NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

GPS SIGNAL PHASE TRACKING USING A KALMAN FILTER

by

Thomas H. Newman

June 1997

Thesis Advisor: Hal Titus

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE 5. GPS SIGNAL PHASE TRACKING USING A KALMAN FILTER			5. FUNDING NUMBERS
6. AUTHOR(S) Thomas H. Newman			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

One of the more significant error sources in using the Global Positioning System for attitude determination of aircraft and earth orbiting satellites is multipath interference. The GPS signal is reflected from the spacecraft structures resulting in a composite signal that differs from the direct line-of-sight signal in terms of phase and amplitude. The resulting phase errors can have a significant impact on the attitude determination error. This thesis will describe the GPS signal generation and provide tools to simulate the signal. A Kalman filter will then be developed to track the phase changes in the simulated GPS signals, and its performance described in the absence and presence of multipath.

14. SUBJECT TERMS .GPS, KALMAN FILTER, MULTIPATH			15. NUMBER OF PAGES 115
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

GPS SIGNAL PHASE TRACKING USING A KALMAN FILTER

Thomas H. Newman
Lieutenant, United States Navy
B.S., University of South Carolina, 1991

Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE IN ENGINEERING SCIENCE
(ELECTRICAL ENGINEERING)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 1997**

Author: [illegible]
Thomas H. Newman

Approved by: Hal Titus
Hal Titus, Thesis Advisor

Vicente Garcia
Vicente Garcia, Second Reader

[illegible]
Herschel H. Loomis, Jr., Chairman
Department of Electrical and Computer Engineering

ABSTRACT

One of the more significant error sources in using the Global Positioning System for attitude determination of aircraft and earth orbiting satellites is multipath interference. The GPS signal is reflected from the spacecraft structures resulting in a composite signal that differs from the direct line-of-sight signal in terms of phase and amplitude. The resulting phase errors can have a significant impact on the attitude determination error. This thesis will describe the GPS signal generation and provide tools to simulate the signal. A Kalman filter was developed to track the phase changes in the simulated GPS signals and its performance described in the absence and presence of multipath.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

I would like to thank Professor Titus for his continued support and sharing his expertise during a time when there were much higher priorities for him and his family. Best wishes to Mrs. Titus.

Special thanks to my wife Beth, whose love, support, and encouragement have been invaluable.

I. INTRODUCTION

A. BACKGROUND

In order for a spacecraft to perform its mission, it must be able to orient itself in relation to either the sun, the earth or both within an inertial reference frame. The primary system used to accomplish this is the spacecraft attitude determination and control subsystem (ADCS). The ADCS uses external references to determine the vehicle's absolute attitude with respect to some fixed inertial reference frame. Traditionally, these external references have included the Sun, the Earth's infrared (IR) horizon, the local magnetic field direction, and the stars. Recent technological developments have added a new tool to the external reference arsenal - the Global Positioning System (GPS).

Typical traditional ADCS sensor characteristics are shown below in Table 1. (Larson and Wertz)

SENSOR	TYPICAL PERFORMANCE RANGE	WEIGHT (KG)	POWER (W)
Inertial measurement unit (gyros & accelerometers)	Gyro drift rate = 0.003°/hr to 1°/hr	3 to 25	10 to 200
Sun Sensors	Accuracy=0.005° to 3°	0.5 to 2	0 to 3
Star Sensors (scanners & mappers)	0.0003° to 0.01°	3 to 7	5 to 20
Horizon Sensors	0.1° to 1°	2 to 5	5 to 10
Magnetometer	0.5° to 3°	0.6 to 1.2	< 1

Table 1 Typical ADCS Sensors

In comparison, the Trimble TANS Vector attitude determination system using GPS is capable of an accuracy of 0.1° with a mass of 2.2 kg (GPS receiver and antennas) and a power requirement of 7.5 W. Additionally, the Trimble system also provides three-dimensional orbital position information; which other sensors are incapable of providing.

As described by Cohen (1996), attitude determination using GPS is achieved by determining the relative positions of multiple antennas, mounted to the vehicle, based on differences in the carrier phase measurements at the different antennas. The basic premise is that the GPS satellite is so distant, relative to the receiving antenna separations, that arriving wavefronts can be considered as effectively planar as shown below in Figure 1.

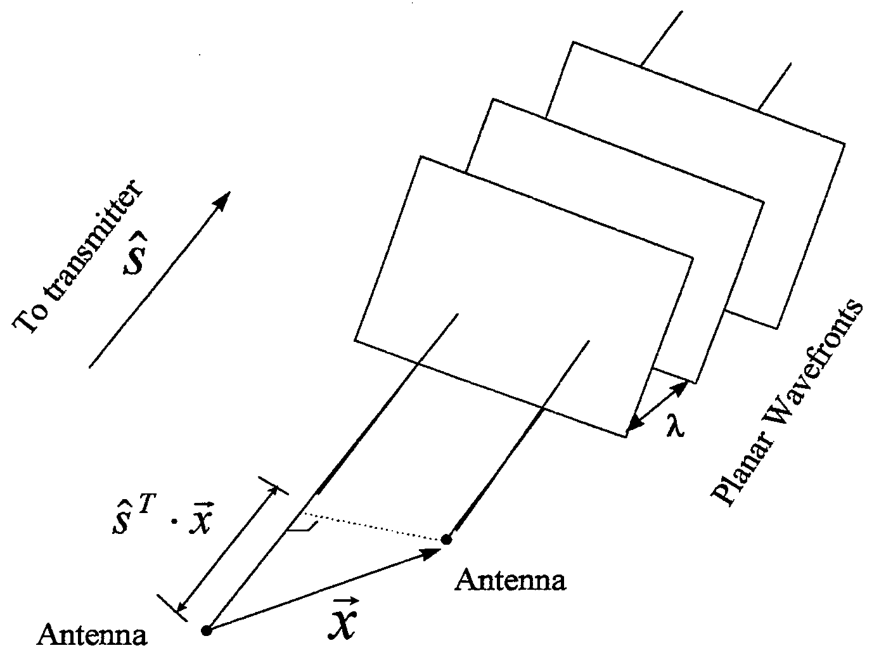


Figure 1 Attitude Geometry

The phase of a signal, ϕ, is a function of the wavelength, λ, and distance traveled, d:

$$\phi = 2\pi \frac{d}{\lambda}$$

The wavelength is a function of the speed of light, c, and the frequency, f. The distance traveled is a function of the speed of light and the time difference of arrival between antennas, Δt:

$$\lambda = \frac{c}{f}$$
$$d = c\,\Delta t$$

Therefore a relationship between, time and phase can be shown to be

$$\varphi = 2\pi \tfrac{x}{\lambda} = 2\pi \frac{c\,\Delta t}{c\,/\,f} = 2\pi f\,\Delta t$$

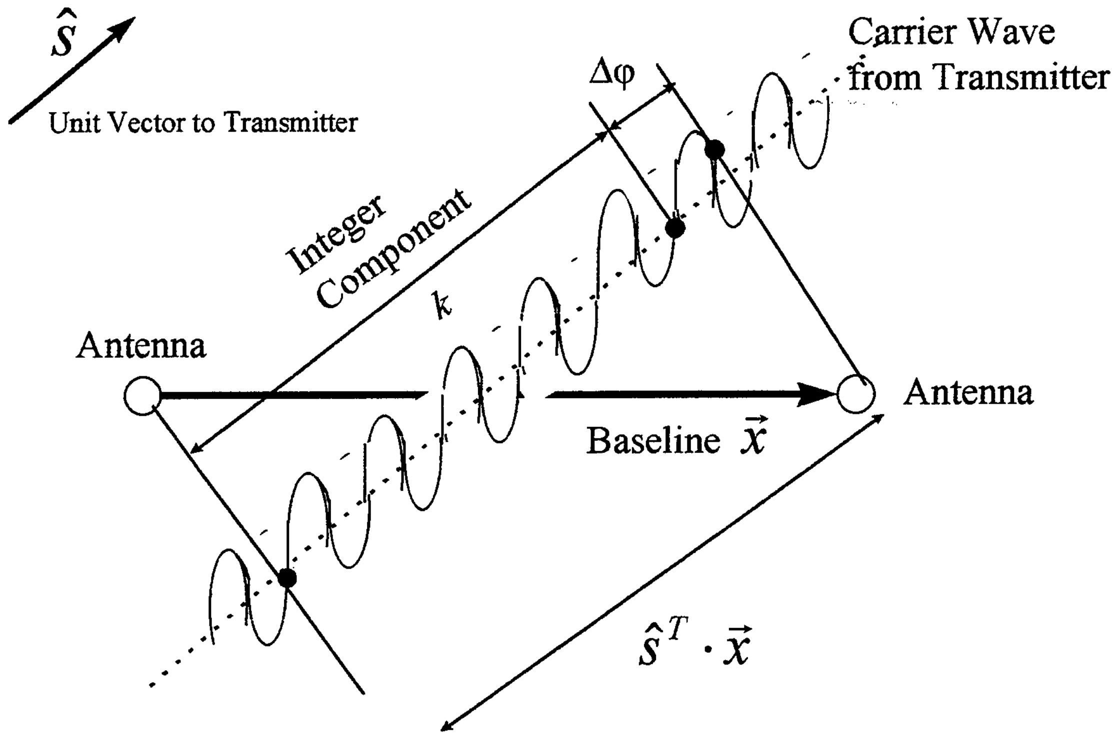
The incoming signal will arrive at the antenna closest to the transmitter before reaching the most distant antenna. By measuring the difference in carrier phase between the antennas, a receiver can determine the relative range between any two antennas. The measured differential phase, $\Delta\varphi$, for baseline *i* and satellite *j*, is proportional to the dot product of the baseline vector, $\vec{x}$, and the line of sight unit vector to the transmitter, $\hat{s}$. As shown in Figure 2, the GPS receiver measures only the fractional part of the differential phase. The integer component, *k*, must be resolved by independent means.

The differential phase can therefore be expressed as:

$$\Delta\varphi_{ij} = \hat{s}^T \cdot \vec{x} - k_{ij} + v_{ij}$$

where v_{ij} is the noise in the measurement. Given the differential phase measurements between the different antennas, the spacecraft attitude can then be resolved as shown by Cohen (1995). Filtering this measurement will be the focus of this thesis.

The dominant noise source in differential phase measurements is multipath, which occurs when the signal arriving at the antenna consists of the reflected signals in addition to the line-of-sight source. The reflected signal is phase shifted with respect to the original transmission and appears as additive noise at the antenna. Because the antenna locations are different, the multipath signature at each antenna is unique and will not be common to every antenna. Multipath accounts for more than 90% of the total error budget in carrier phase measurements. (Lightsey)



$\hat{s}$
Unit Vector to Transmitter
$\Delta\varphi$
Carrier Wave from Transmitter
Integer Component
k
Antenna
Antenna
Baseline $\vec{x}$
$\hat{s}^T \cdot \vec{x}$

Figure 2 Observation Geometry

B. PURPOSE

This thesis will discuss the generation of the GPS carrier signal which is used in making differential phase measurements. Models will be developed for simulating the multipath environment and the GPS signal. A Kalman filter which tracks the phase in the signal for use in the differential phase measurements will be described. Finally, results of the Kalman filter's use in a multipath environment will be presented.

Chapter II will discuss the GPS Signal Generation and present a model of the signal. The multipath interference will be examined and modeled in Chapter III. Chapter IV provides the derivation of the Kalman filter for tracking the phase of a GPS signal. Chapter V will show the results when the filter is used in tracking the GPS phase. The final chapter will summarize the findings and present other ideas for minimizing multipath and suggestions for future investigation.

II. GPS SIGNAL GENERATION

The GPS signal is generated as shown in Figure 3. The GPS satellite transmits two carrier frequencies, L1 (primary frequency) and L2 (secondary frequency). These carrier frequencies are modulated by spread spectrum codes with a pseudorandom noise (PRN) sequence unique to each satellite. The L1 frequency is modulated by the navigation message along with two PRN codes: the coarse/acquisition (C/A) code and the precision (P) code. The GPS L2 signal is modulated by one of the codes using Binary Phase Shift Keying (BPSK). The GPS L1 signal is modulated using both codes by an unbalanced quadrature phase shift key (QPSK) modulator which is actually two BPSK modulators with different amplitudes. This development closely follows that given by Kaplan (1996).

A. PN CODE GENERATION

The PRN codes used in the generation of the GPS signal are non-maximum length codes also known as Gold codes. These codes are generated from an n-bit shift register generator. This register has an initial state or fill. After each clock cycle the output of a specified cell is used as the input to a modulo-2 adder (exclusive-or). The output of the modulo-2 adder is then fed back into a specified stage of the register. As an example, consider the shift register shown in Figure 4. This register uses the 3rd and 5th stages as the input to the modulo-2 adder and feeds the result back into the first stage. The polynomial used to describe the design of linear code generators is of the form

$$1+\sum x^i$$

where x^i means the output of the ith cell of the shift register is used as the input of the modulo-2 adder and the 1 means the output of the adder is feed in to stage 1. The polynomial for the register in Figure 4 is then $1 + X^3 + X^5$. If this register was initially filled with all 1's at time 0, the resulting register for the first ten clock cycles would be as shown in Table 2. The binary signal output is taken from the last register and in this case would be 11111000110.

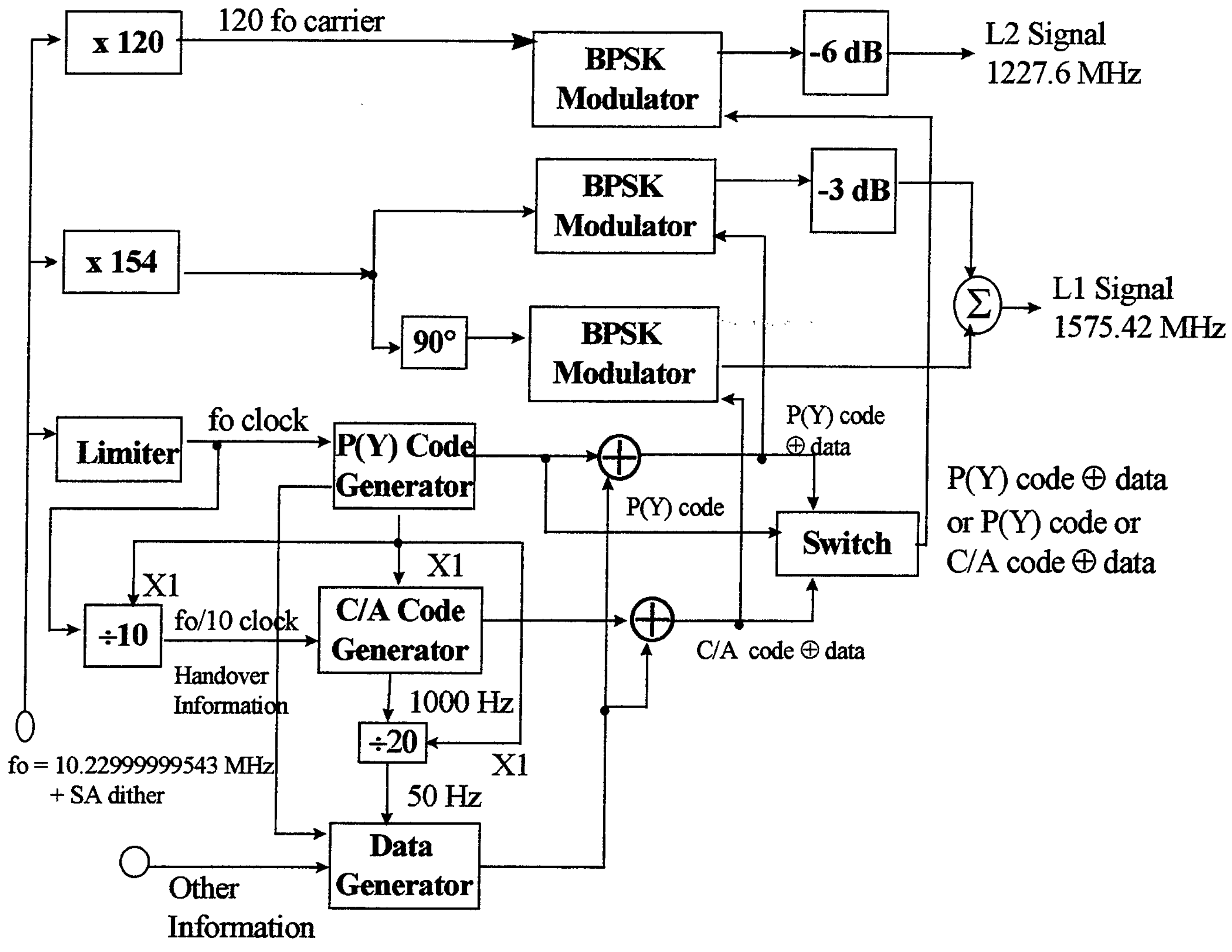


Figure 3 GPS L1 and L2 Signal Generation

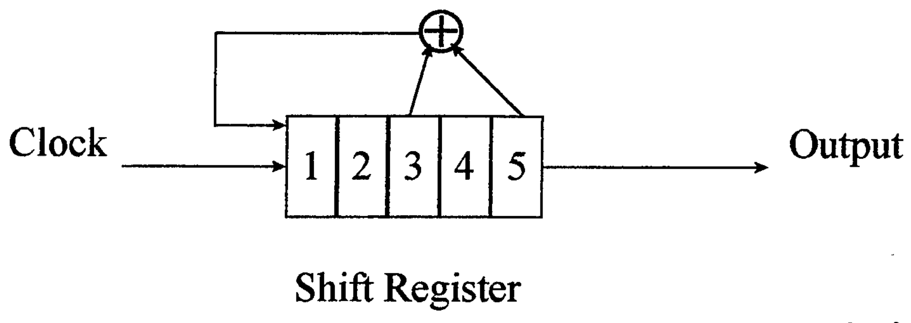


Figure 4 PRN Code Shift Register for Example Polynomial $1+x^3+x^5$

	Register				
time	**1**	**2**	**3**	**4**	**5**
0	1	1	1	1	1
1	0	1	1	1	1
2	0	0	1	1	1
3	0	0	0	1	1
4	1	0	0	0	1
5	1	1	0	0	0
6	0	1	1	0	0
7	1	0	1	1	0
8	1	1	0	1	1
9	1	1	1	0	1
10	0	1	1	1	0

Table 2 PRN Code Generation Example Results for Polynomial $1+x^3+x^5$

The Matlab m-function *gen poly.m* can be used to generate the PRN sequence for any user defined polynomial and register.

B. C/A CODE GENERATION

The GPS coarse/acquisition (C/A)-code is a Gold code generated by two 10-bit shift registers, G1 and G2. The polynomial for each is given below (Kaplan, 1996)

$$G1 = 1 + X^3 + X^{10}$$

$$G2 = 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}$$

Both C/A-code registers have an initial fill of all 1's. Each code is derived and the result of G2 is then delayed by the satellite PRN code number. This delay effect is obtained by the exclusive-or of the selected positions of the G2 register. The C/A-code is then obtained by the exclusive-or of the G1 and delayed G2 signals. An example for satellite vehicle 1 is shown in Figure 5.

The code tap positions and initial code sequences are given in Table 3. The m-files *g1.m* and *g2.m* generate the codes for a specified satellite vehicle number.

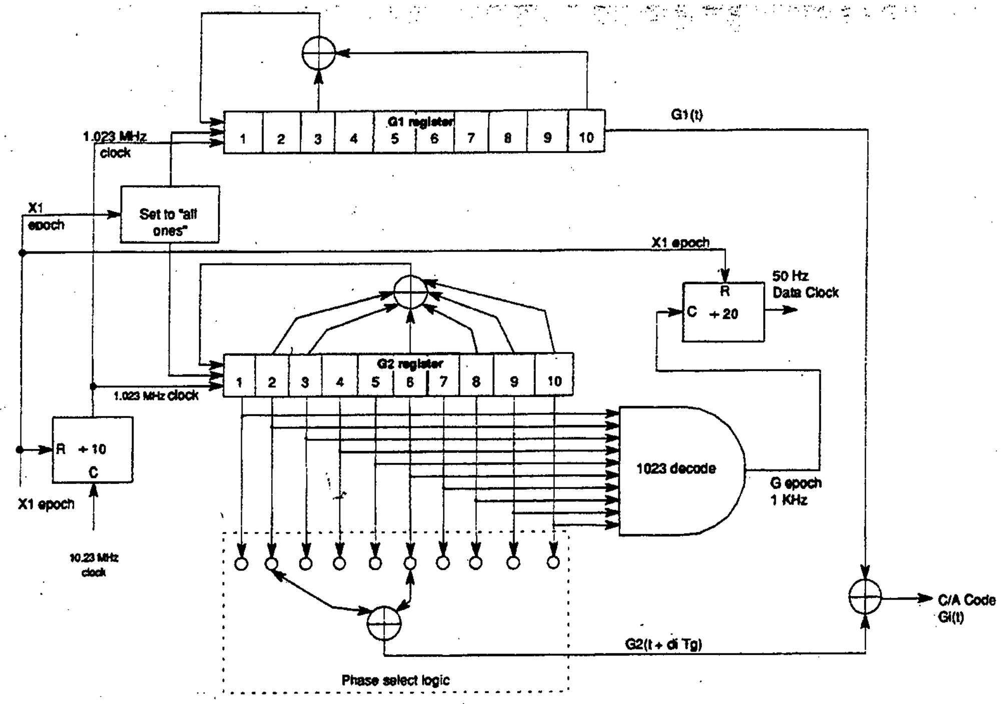


Figure 5 C/A-Code Generator for Satellite Vehicle 1 with C/A Code Tap Selection 2⊕6 (from Kaplan)

SV PRN Number	C/A-Code Tap Selection	C/A-Code Delay (Chips)	P-Code Delay (Chips)	First 10 C/A Chips (Octal)	First 12 P-Chips (Octal)
1	2⊕6	5	1	1440	4444
2	3⊕7	6	2	1620	4000
3	4⊕8	7	3	1710	4222
4	5⊕9	8	4	1744	4333
5	1⊕9	17	5	1133	4377
6	2⊕10	18	6	1455	4355
7	1⊕8	139	7	1131	4344
8	2⊕9	140	8	1454	4340
9	3⊕10	141	9	1626	4342
10	2⊕3	251	10	1504	4343
11	3⊕4	252	11	1642	4343
12	5⊕6	254	12	1750	4343
13	6⊕7	255	13	1764	4343
14	7⊕8	256	14	1772	4343
15	8⊕9	257	15	1775	4343
16	9⊕10	258	16	1776	4343
17	1⊕4	469	17	1156	4343
18	2⊕5	470	18	1467	4343
19	3⊕6	471	19	1633	4343
20	4⊕7	472	20	1715	4343
21	5⊕8	473	21	1746	4343
22	6⊕9	474	22	1764	4343
23	1⊕3	509	23	1063	4343
24	4⊕6	512	24	1706	4343
25	5⊕7	513	25	1743	4343
26	6⊕8	514	26	1761	4343
27	7⊕9	515	27	1770	4343
28	8⊕10	516	28	1774	4343
29	1⊕6	859	29	1127	4343
30	3⊕7	860	30	1453	4343
31	3⊕8	861	31	1625	4343
32	4⊕9	862	32	1712	4343

Table 3 GPS Code Phase Assignments (from GPS-ICD-2000)

test_CA.m is a script file to test C/A code generation. This program calculates the first 10 C/A-chips and converts the result to octal numbers that match the First 10 C/A-Chips (Octal) of the GPS-ICD-2000 table shown in Table 3

For example the first 10 chips of the PRN 5 C/A-code is 1001011011 which is 1133 in octal.

C. P-CODE GENERATION

The GPS Precision (P) Code is generated by PRN sequences using four 12 bit shift registers whose polynomials and initial states are shown below:

Register	Polynomial	Initial State
X1A	$1 + X^1 + X^2 + X^5 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$	001001001000
X1B	$1 + X^1 + X^2 + X^3 + X^4 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$	010101010100
X2A	$1 + X^1 + X^3 + X^4 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$	100100100101
X2B	$1 + X^2 + X^3 + X^4 + X^8 + X^9 + X^{12}$	010101010100

Table 4 P-Code Generator Polynomials and Initial States

The P-code is 7 days in length at a chipping rate of 10.23 MBps. The 7 day sequence is the Modulo-2 sum of two subsequences X1 and $X2_I$; their lengths are 15,345,000 chips and 15,345,037 chips respectively.

X1 itself is generated by the Modulo-2 sum of the output of two 12-stage registers (X1A and X1B) short cycled to 4092 and 4093 chips respectively. When the X1A short cycles are counted to 3750 the X1 epoch is generated. This occurs every 1.5 seconds, after 15,345,000 chips of the X1 pattern have been generated. The X1 period is defined as 3750 X1A cycles (15,345,000 chips) which is not an integer number of X1B cycles. To accommodate this the X1B shift register is held in the final state (chip 4093) of its 3749th cycle for 343 chips. The X2 sequence is generated in a similar manner, however the X2 period is defined to be 37 chips longer than the X1 period in order to force the X2A and

X2B epochs to precess with respect to the X1a and X1B epochs. This precession continues until the end of the 7 day period.

The Y-code[1] is used when the anti-spoofing mode of operation is activated. It is generated in the same fashion as the P-code, however the polynomial generating equations are classified. (ICD-GPS-2000)

x1a.m, x1b.m, x2a.m and *x2b.m* are used to calculate the PN codes. The m-file *test_P.m* is used to test P-code generation.

D. DATA GENERATOR

The output of the data generator is the navigation message, $D(t)$. The message includes satellite vehicle ephemerides, system time, clock bias information, status messages, and C/A to P-code handover information. The navigation message is generated at 50 bps and is added to the P and C/A-codes to modulate both the L1 and L2 signals. (ICD-GPS-200)

E. BPSK MODULATOR

After the various codes are generated they are then used to modulate the carrier signal using binary phase shift keying, BPSK, resulting in the L1 or L2 signal. The Simulink program used to simulate the BPSK modulator is shown below in Figure 6.

[1] The Y-code and P-code are sometimes discussed together and are referred to as P(Y)-code meaning P (or Y) code.

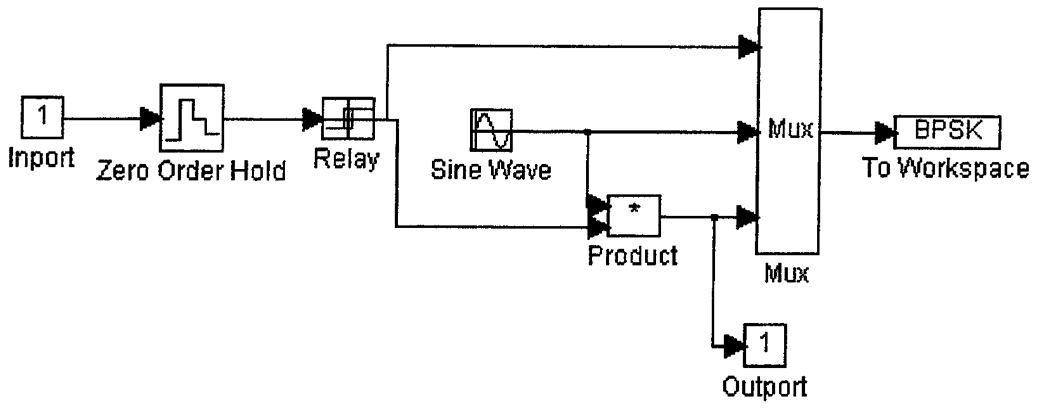


Figure 6   BPSK Simulink Diagram

The PRN code is input via the inport. It is sent through a zero-order-hold and relay function to generate a non-return to zero (NRZ) signal, which is multiplied by the sine wave, resulting in a binary phase shifted signal.

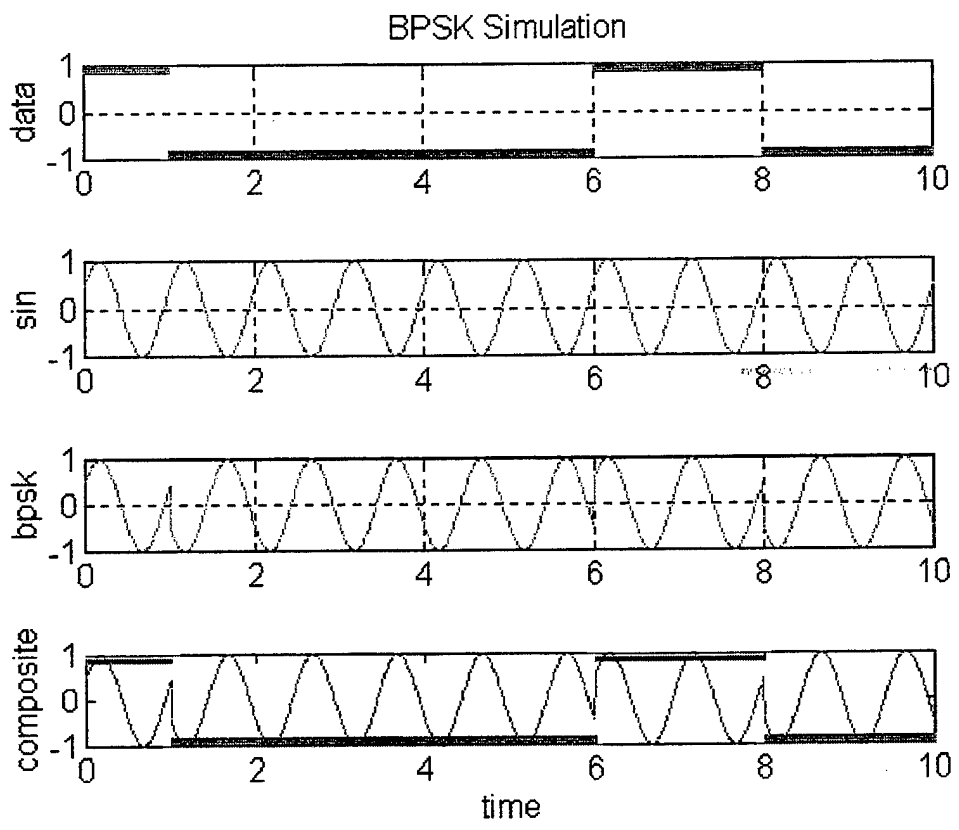


Figure 7   BPSK Simulation Results

Figure 7 shows an example of the BPSK simulation results. This figure shows the results: NRZ signal in the top graph, the sine wave in the second graph, the product of the two, or the BPSK signal in the third graph, and a composite of the data and resulting signal in the bottom graph.

F. L1 SIGNAL

The P-code at, 10.23×10^6 chips per second, and C/A-code, at 1.023×10^6 chips per second, are combined with the 50 bits per second data and are then used to modulate the L1 frequency ($154 \cdot f_0 = 1575.42\,MHz$) as shown in Figure 3. This results in an Unbalanced Quadri-Phase Shift Keying (UQPSK) modulation with the data bits added to the ranging codes, the C/A-code signal lagging the P-code signal by 90°; and the C/A-code signal power nominally exceeding the P-code signal power by 3 dB. (Struza) Mathematically, this can be represented by:

$$L_1(\omega_1 t) = A_1[P(t) \otimes D(t)]\cos(\omega_1 t) + \sqrt{2}A_1[G(t) \otimes D(t)]\sin(\omega_1 t)$$

where $P(t)$ is the P-code, $D(t)$ is the data, and $G(t)$ is the C/A-code.

G. L2 SIGNAL

The L2 frequency ($120 \cdot f_o = 1227.6\,MHz$) is modulated by either the exclusive-or of the P-code and data, the P-code, or the exclusive-or of the C/A code and data, as selected by the control segment. Mathematically, this can be represented by:

$$L_2(\omega_2 t) = A_2[P(t) \otimes D(t)]\cos(\omega_2 t)$$
$$L_2(\omega_2 t) = A_2[P(t)]\cos(\omega_2 t)$$
$$L_2(\omega_2 t) = A_2[G(t)]\cos(\omega_2 t)$$

H. SUMMARY

The GPS signal consists of two separate signals, L1 and L2. These signals are generated using BPSK and unbalanced QPSK modulation. The modulation signal is generated using the P and C/A codes to which the data signal is added. The Matlab files *init.m, sig gen.m* and *my glb.m* are used to initialize and generate a sample of the codes used to modulate the signal. These codes are then used by the Simulink programs *L1 sim*.m and *L2 sim*.m to generate a sample of the signal. In the next chapter the multipath environment will be described.

III. MULTIPATH

A. INTRODUCTION

This chapter will discuss the multipath environment that impacts the use of GPS for attitude determination and will present a method to model multipath interference using Matlab.

Multipath occurs when the signal arrives at the antenna from reflected surfaces and from the line of sight sources. A simple case with a single reflected path signal is shown in Figure 8.

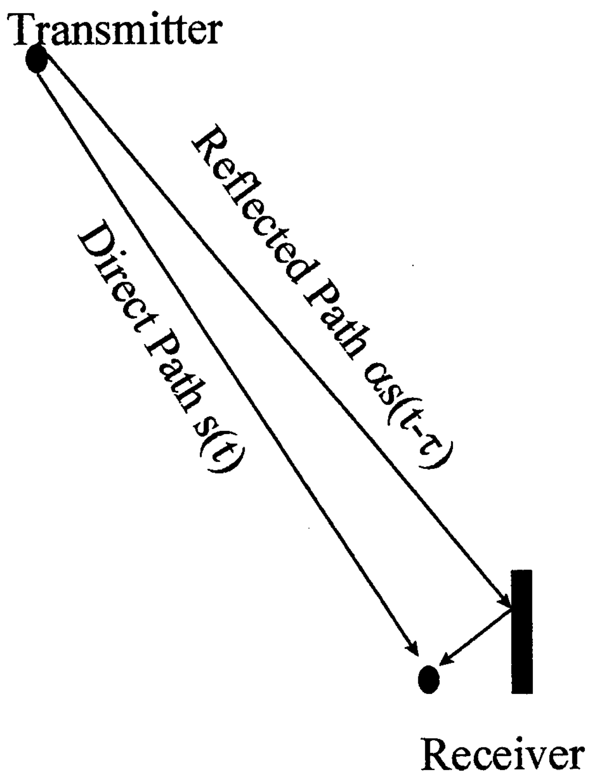


Figure 8 Simple Multipath Example

The reflected signal is phase shifted with respect to the original transmission and appears as additive noise at the antenna. When antenna locations are different, the multipath signature at each antenna is unique and the error in the phase measurement is not common. (Lightsey) Multipath interference, due to environmental factors, will impact the signal, however this effect will be common to each received signal.

For the developed model it will be assumed that the multipath environment will be static. In reality the multipath environment will be varying due to the satellites attitude and position relative to the GPS satellites changing over time.

B. MATHEMATICAL MODEL

In the absence of the multipath, the input signal to a GPS receiver for the L1 QPSK signal is given by:

$$s(t) = A_c \cos(\omega t + a(t)\tfrac{\pi}{2})$$

where A_c is the received signal amplitude and a(t) = 0,1,2 or 3 is the pseudo-random code waveform that phase modulates the carrier. (Kumar) In the presence of N multipaths, in addition to the direct line of sight path, the input signal is given by

$$s_m(t) = \alpha_0 s(t) + \alpha_1 s(t - \tau_1) + \ldots + \alpha_N s(t - \tau_N)$$

where α_i and τ_i denote the amplitude and delay of the i$^{\text{th}}$ multipath for i=1,2,...,N. Graphically, this can be represented as shown in Figure 9.

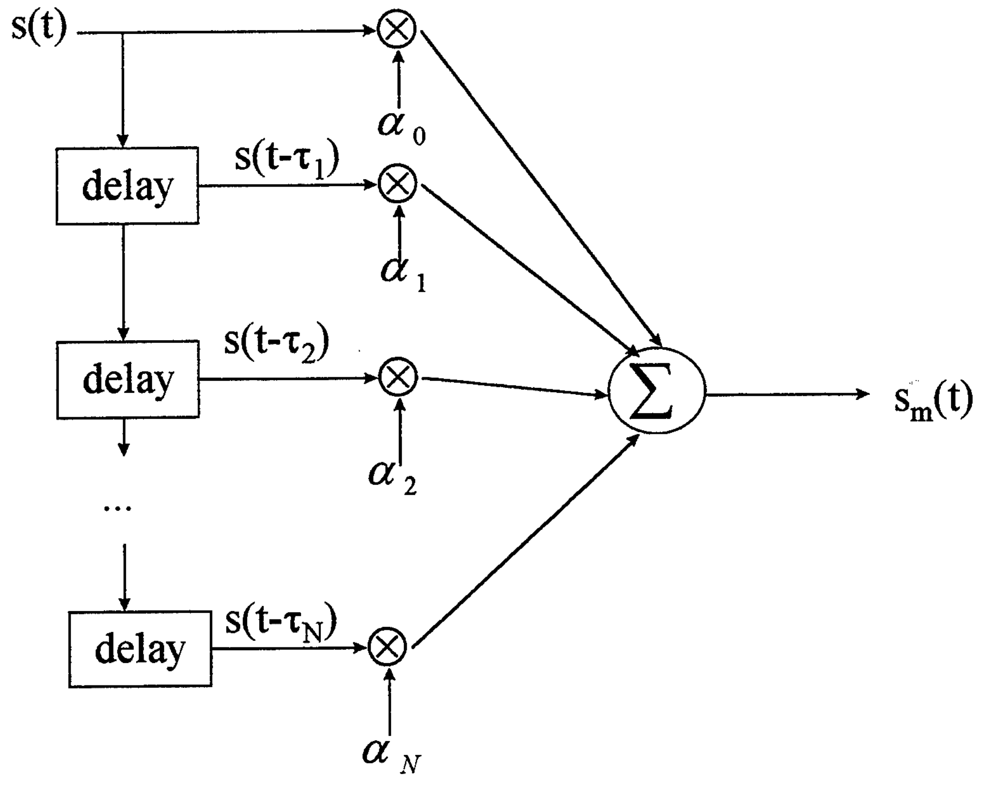


Figure 9 Multipath Block Diagram

Typically for BPSK and QPSK signals the carrier frequency operates at a much higher frequency than the modulation rate. As a result, a small value of delay, τ, will result in large phase differences. When the multipath signal with a large phase difference is added to the direct path there may be destructive adding which can significantly impact the received signal.

C. SIMULATION MODEL

In the model *mpath.m,* three vectors are used to develop the signal s_m

delay -1 by τ_N vector of delayed samples. The first entry is the undelayed signal, the next entry is the signal delayed by one sample period. The length of this vector is the maximum value in the tau vector and is initialized as all zeros.

tau - vector of delay times. Each entry is 1 minus the delay time in sample periods, dt, and is used to index the delay vector. The first entry is 1 and represents a delay of zero.

alpha - vector of amplitudes. Each entry is the amplitude of the corresponding delayed signal. The first entry is a 1 in order to return the undelayed line of sight signal.

As an example, the BPSK signal shown in Figure 10, with N=3 multipath signals, will be used to demonstrate the use of the vectors and the algorithm used to develop the signal

$$s_m(t) = \alpha_0\, s(t-\tau_0) + \alpha_1 s(t-\tau_1) + \alpha_2\, s(t-\tau_2) + \alpha_3 s(t-\tau_3)$$

where

$$\tau = \begin{bmatrix} 1 & 10 & 12 & 20 \end{bmatrix}$$
$$\alpha = \begin{bmatrix} 1 & 0.5 & 0.3 & 0.1 \end{bmatrix}$$

thus

$$s_m(t) = s(t) + 0.5\cdot s(t-9\cdot dt) + 0.3\cdot s(t-11\cdot dt) + 0.1\cdot s(t-19\cdot dt)$$

If the sample period, dt, equals 0.01 the following signal is produced using the m-file *mpath.m*

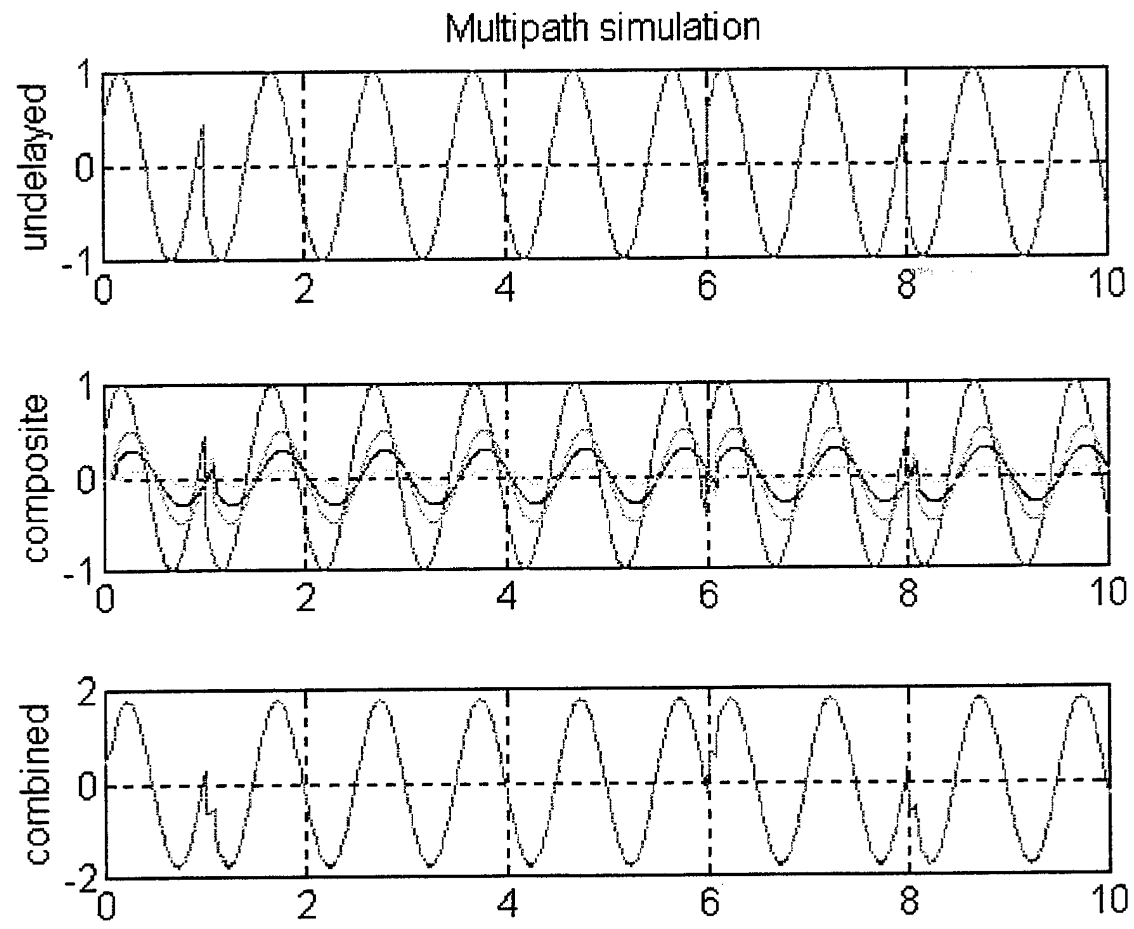


Figure 10   Multipath Example Signal

The top graph in Figure 10 shows the undelayed signal, the middle graph shows the direct path and multipath signals plotted individually and the bottom graph shows the sum of the individual and multipath signals.

If the multipath delay time and amplitude are increased with the following parameters

$$\tau = [1 \quad 20 \quad 22 \quad 30]$$
$$\alpha = [1 \quad 0.5 \quad 0.3 \quad 0.5]$$

the resulting multipath signal will be as shown here:

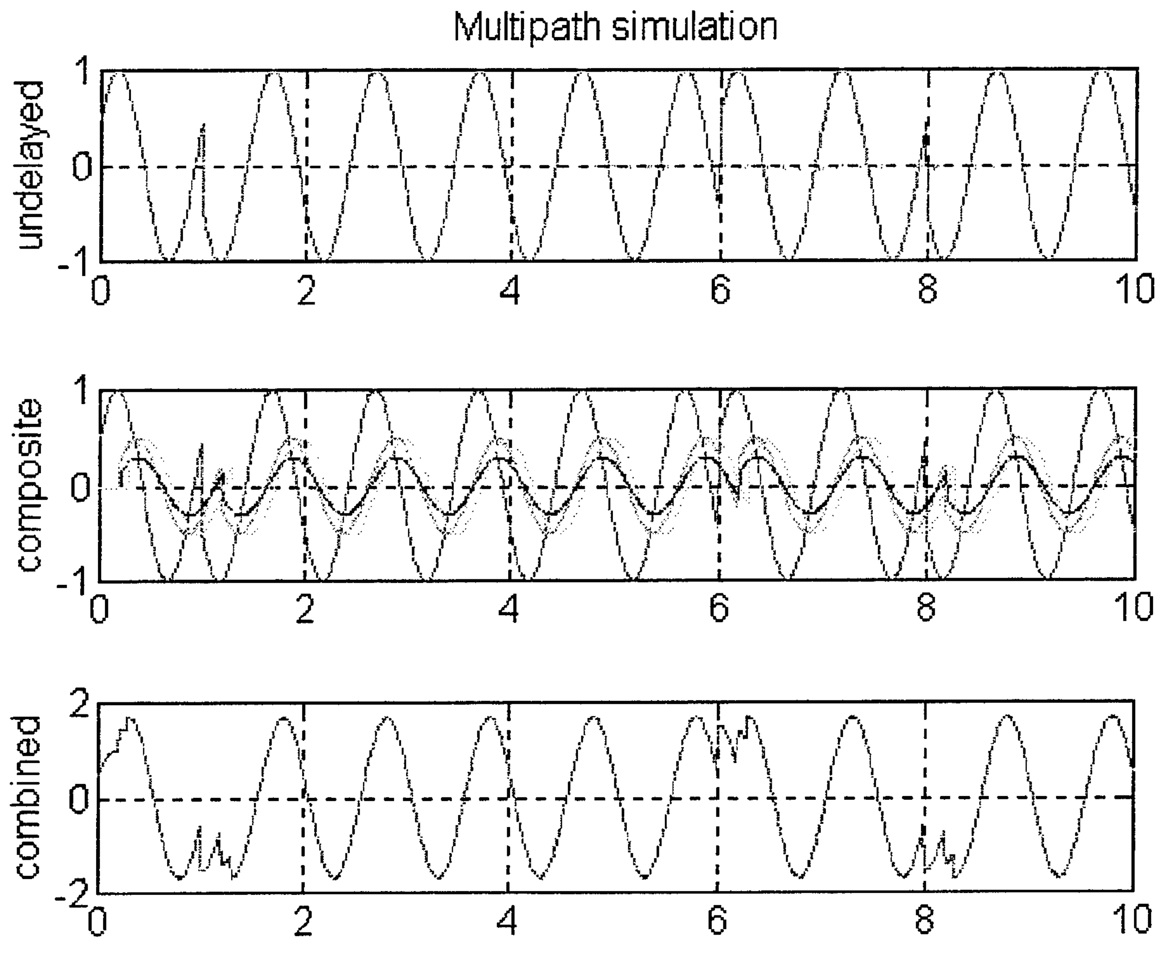


Figure 11   Multipath Example with Large Delay

Notice that in both examples the multipath signal has a different amplitude from the original signal and that the larger the delay the larger the change in the phase of the signal. Below is an example of destructive interference obtained with the values

$$\tau = \begin{bmatrix} 1 & 50 & 52 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 1 & 0.5 & 0.4 \end{bmatrix}$$

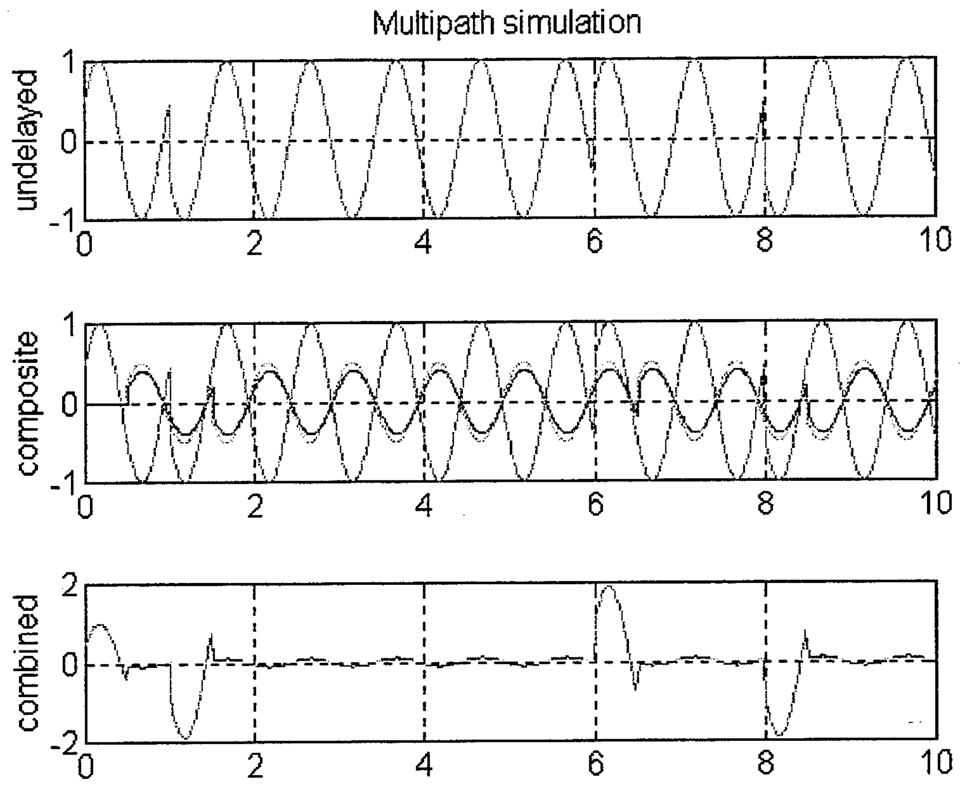


Figure 12 Multipath Example with Destructive Interference

This chapter has described the multipath environment and presented a simulation model to generate a signal based on user defined parameters. The next chapter will describe a Kalman filter which is used for tracking the GPS signal.

IV. KALMAN FILTER FOR TRACKING PHASE

A. DISCRETE KALMAN ESTIMATOR

The development of a filter to track the GPS signal in a multipath environment began with the implementation of a Discrete Kalman Estimator for tracking the phase changes of BPSK signal. The derivation below closely follows that given by Professor Kirk (1975).

1. Plant Equations

The plant equations were derived as follows:

$x_1 = \cos(\omega t)$

$x_2 = \dot{x}_1 = -\omega \sin(\omega t)$

$$\mathbf{x}(t) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \cos(\omega t) \\ -\omega \sin(\omega t) \end{pmatrix}$$

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -\omega \sin(\omega t) \\ -\omega^2 \cos(\omega t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos(\omega t) \\ -\omega \sin(\omega t) \end{pmatrix} = \mathbf{A}\mathbf{x}(t)$$

2. Plant Characterization

The discrete model of the plant is characterized by the linear discrete difference equations:

$$x(k+1) = \phi\, x(k) + w(k)$$

$$z(k) = Cx(k) + v(k)$$

where:

x(k) is the n-dimensional state vector at time k,

z(k) is a q-dimensional output measurement vector at time k

w(k) is a p-dimensional vector of random forcing inputs at time k

ϕ is the state transition matrix

C is the observation matrix

and

v(k) is the q-dimensional vector of random measurement noise at time k

3. Gain, and Covariance Equations

The gain, G, and covariance equations derived by Kirk (1975) are:

$$G(k) = P(k|k-1)\cdot C^T \cdot \left[C \cdot P(k|k-1)\cdot C^T + R\right]^{-1}$$

$$P(k|k) = [I - G(k)\cdot C]\cdot P(k|k-1)$$

$$P(k+1|k) = \phi \cdot P(k|k)\cdot \phi^T + Q$$

where

C is the observation matrix

I is the identity matrix,

G is the Kalman gain

P is the variance of estimation error

Q is random process forcing input covariance matrix

R is the variance of measurement noise.

The estimator equations are: $\hat{x}(k|k) = \hat{x}(k|k-1) + G(k)\cdot[z(k) - C\cdot\hat{x}(k|k-1)]$ [2]

$$\hat{x}(k|k-1) = \phi\cdot\hat{x}(k-1|k-1) + \Delta\cdot u(k-1)$$

with the initial condition $\hat{x}(0|-1) = \overline{x_o}$

[2] $\hat{x}(j|j)$ is read as x hat of j given j and is defined as the estimate of x at time j given measurements up to and including time j.

4. Assumptions

a) The measurement noise has zero mean:

$$E[v(k)] = 0 \quad k = 0,1,\ldots$$ [3]

is uncorrelated and has covariance R(k):

$$E[v(k)v^T(j)] = E[v(j)v^T(k)] = R(k)\delta_{kj}$$

where δ_{jk} is the Kronecker delta function defined by

$$\delta_{jk} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

b) The initial state is a random variable with known mean

$$E[x(0)] = \overline{x_o}$$

and covariance:

$$E\left[\left[x(0) - \overline{x_o}\right]\left[x(0) - \overline{x_o}\right]^T\right] = M$$

The measurement noise and initial state are uncorrelated:

$$E[x(0)v^T(k)] = E[v(k)x^T(0)] = 0 \quad k = 0,1,\ldots$$

c) by the linear relationship

$$\hat{x}(k|k) = \hat{x}(k|k-1) + G(k)\cdot[z(k) - C\cdot\hat{x}(k|k-1)]$$

where G(k) is the Kalman gain and $z(k) - C\cdot\hat{x}(k|k-1)$ is the correction term, or the difference between the measured and predicted value.

d) The random process forcing input has zero mean

$$E[w(k)] = 0 \quad k = 0,1,\ldots$$

is uncorrelated and has covariance Q(k)

$$E[w(k)w^T(j)] = E[w(j)w^T(k)] = Q(k)\delta_{kj} \quad j,k = 0,1,2,\ldots$$

e) The forcing random process and initial state value are uncorrelated

[3] $E[\cdot]$ is the expected value or mean of the random variable.

$$E\left[w(k)x^T(0)\right] = E\left[x(0)w^T(k)\right] = 0 \quad k = 0,1,2,\ldots$$

f) The forcing random process and measurement noise process are uncorrelated

$$E\left[w(k)v^T(j)\right] = E\left[v(j)w^T(k)\right] = 0 \quad k = 0,1,2,\ldots$$

B. COMPUTATIONAL PROCEDURE

The computational procedure is shown in the following algorithm.

```
Initialize with
    k=0
    $\hat{x}(0|-1) = \overline{x_o}$
    $P(k|k-1) = M$
The procedure iterates on k in the loop:
    $G(k) = P(k|k-1)\cdot C^T \cdot \left[C \cdot P(k|k-1)\cdot C^T + R\right]^{-1}$
    $P(k|k) = [I - G(k)\cdot C]\cdot P(k|k-1)$
    $P(k+1|k) = \phi \cdot P(k|k)\cdot \phi^T + Q$ {this will be $P(k|k-1)$ next time through}
    take measurement $z(t)$
    $\hat{x}(k|k) = \hat{x}(k|k-1) + G(k)\cdot[z(k) - C\cdot \hat{x}(k|k-1)]$
    $\hat{x}(k|k-1) = \phi \cdot \hat{x}(k-1|k-1) + \Delta \cdot u(k-1)$
    k=k+1
End of the Loop
```

C. PHASE TERMS

The QPSK signal can be represented by

$s(t) = A\cos(\omega t + \phi(t))$

where $\phi(t)$ represents the phase change of the signal

$$\phi(t) = \begin{cases} \phi_o \\ \phi_o + \dfrac{\pi}{2} \\ \phi_o + \pi \\ \phi_o + \dfrac{3\pi}{2} \end{cases}$$

Substituting in these phase values and simplifying using trigonometric identities results in

$$s(t) = \begin{cases} A\cos(\omega t + \phi_o) \\ -A\sin(\omega t + \phi_o) \\ -A\cos(\omega t + \phi_o) \\ A\sin(\omega t + \phi_o) \end{cases}$$

Using these relationships and the sine and cosine information in the state estimates, the Kalman filter can estimate which phase change has occurred. The error term is calculated by finding the estimate of the signal that minimizes the difference between the measured value. Assuming

$$\hat{\mathbf{x}}(k|k-1) = \begin{pmatrix} \cos(\omega t + \phi_o) \\ -\omega\sin(\omega t + \phi_o) \end{pmatrix}$$
$$\mathbf{C} = [1,0]$$
$$\mathbf{C}_{ps} = [0,1]$$

the four error terms can be calculated as shown below, and the minimum value is used to update the prediction

$$e_0 = z(k) - \mathbf{C}\mathbf{x}(k|k-1) = z(k) - \cos(\omega k + \phi_o)$$
$$e_1 = z(k) + \mathbf{C}_{ps}\mathbf{x}(k|k-1)/\omega = z(k) - \sin(\omega k + \phi_o)$$
$$e_2 = z(k) + \mathbf{C}\mathbf{x}(k|k-1) = z(k) + \cos(\omega k + \phi_o)$$
$$e_3 = z(k) - \mathbf{C}_{ps}\mathbf{x}(k|k-1)/\omega = z(k) + \sin(\omega k + \phi_o)$$

These error terms are then used to update the estimate of the signal without any phase shifting which will be called y, and an estimate of the phase shifted signal called y_ps.

Next, the results of using the developed Kalman filter to track the phase of a BPSK, QPSK, and unbalanced QPSK signal will be shown.

V. INITIAL RESULTS

This chapter will show the results obtained when the Kalman filter, discussed in the previous chapter, is used the track the phase of a BPSK, QPSK and unbalanced QPSK[4] signal. The performance of the Kalman filter, will be shown with and without noise, and in the presence of multipath interference. It will be shown that the Kalman filter will track the combined line of sight and direct path signals phase and amplitude. It will then be shown that when given *a priori* information the filter can distinguish between the line of sight and multipath signals and correctly track them, given a static multipath environment.

A. SIGNAL GENERATION

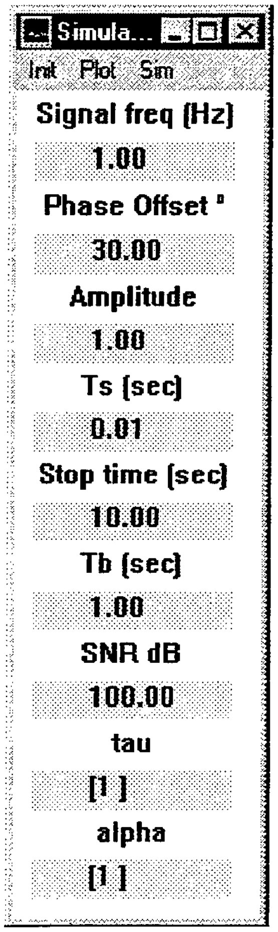
The Matlab m-file *ui_vars.m* sets up the user interface shown in Figure 13 that can be used to control the simulation.

Each entry can be edited by the user to change each of the parameters discussed below:

- *Signal freq* - Sets the frequency, of the oscillator, in Hertz, used by the carriers in the BPSK and QPSK signal generation.
- *Phase Offset* - Determines the initial phase offset, of the line of sight signal in degrees.
- *Amplitude* - Sets the amplitude of the generated signal.
- *Ts* - This sets the sampling period in seconds. In the example below Ts =0.01, resulting in 100 samples per second.
- *Stop time* - Sets the Simulink parameter stop time to control length of simulation.
- *Tb* - Sets the bit period which determines the length of time for zero-order hold per data bit.

[4] An unbalanced QPSK signal is a QPSK signal with differing energy in the in-phase and quadrature phase components. For GPS the L1 signal the amplitude of the quadrature term is equal to $\sqrt{2}$ times the in phase term.

- *SNR dB* - Using this value and the entered amplitude value of the signal, the random number generator is multiplied by a scaling factor to simulate the desired SNR level.
- *tau* - Delay vector used to control the multipath delay times. See Chapter III Section C for more information.
- *alpha* - Amplitude vector - Used to control the multipath amplitudes.



Signal freq (Hz)
1.00
Phase Offset °
30.00
Amplitude
1.00
Ts (sec)
0.01
Stop time (sec)
10.00
Tb (sec)
1.00
SNR dB
100.00
tau
[1]
alpha
[1]

Figure 13 User Interface to Control BPSK and QPSK Simulation

The pulldown menu *Init* initializes all the entered values. *Plot* gives the user a choice of plots to choose from to document the output of the simulation. *Sim* allows the user to choose which simulation will be run.

B. BPSK

Using Matlab and Simulink, a BPSK signal was generated and the algorithm discussed in the previous chapter was used to track the phase changes in the signal.[5]

1. BPSK Signal Without Noise

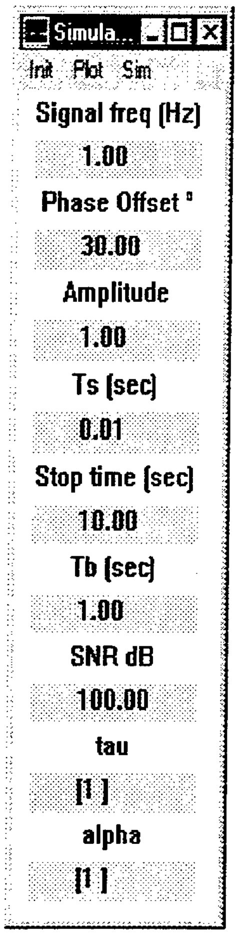
Figure 14 shows the performance of the algorithm when there is no noise present. In this graph, x is the plant state, z is the measured signal (x +noise), y is the recovered carrier, and y_ps is the recovered phase shifted signal. The last graph shows the error in the recovered signal which is the difference in the generated signal and recovered estimate, x-y_ps. Notice there is an initial error during initialization, however the error quickly goes to zero as expected in the absence of noise.

2. With Noise

Keeping all variables the same as above and adding zero mean Gaussian white noise, by using the random function in Matlab, with an SNR value of 12 dB[6], the carrier is still recovered, however small errors are now introduced in the estimate of the phase shifted signal.

[5] See Matlab m-files *kal_init.m, kalman2.m* and the Simulink program *kal_sim*

[6] Spilker states that typical values for C/N_0 are 40.6 dB-Hz, which corresponds to an SNR of 25 dB. See Appendix C for a link budget estimate of the signal to noise ratio.



Signal freq (Hz)
1.00
Phase Offset °
30.00
Amplitude
1.00
Ts (sec)
0.01
Stop time (sec)
10.00
Tb (sec)
1.00
SNR dB
100.00
tau
[1]
alpha
[1]

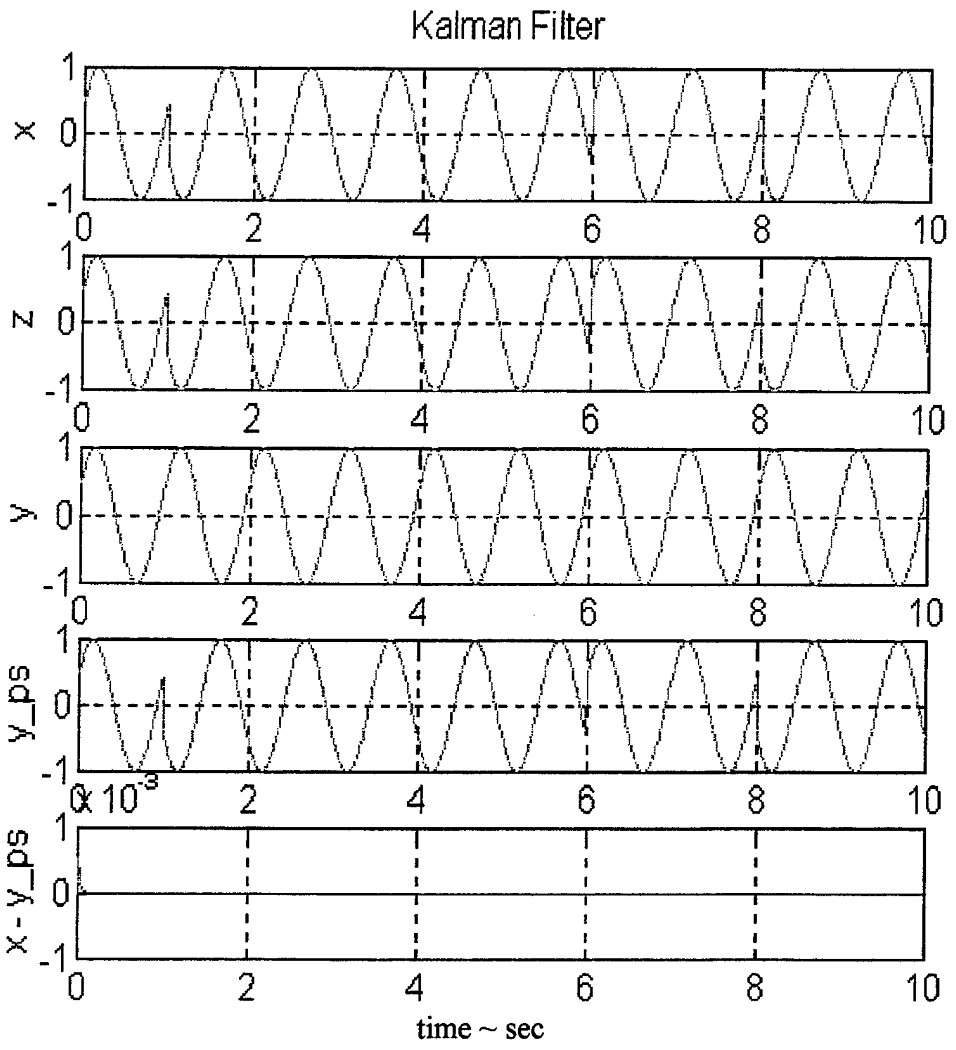


Figure 14   Kalman Filter as Phase Lock Loop for BPSK Without Noise

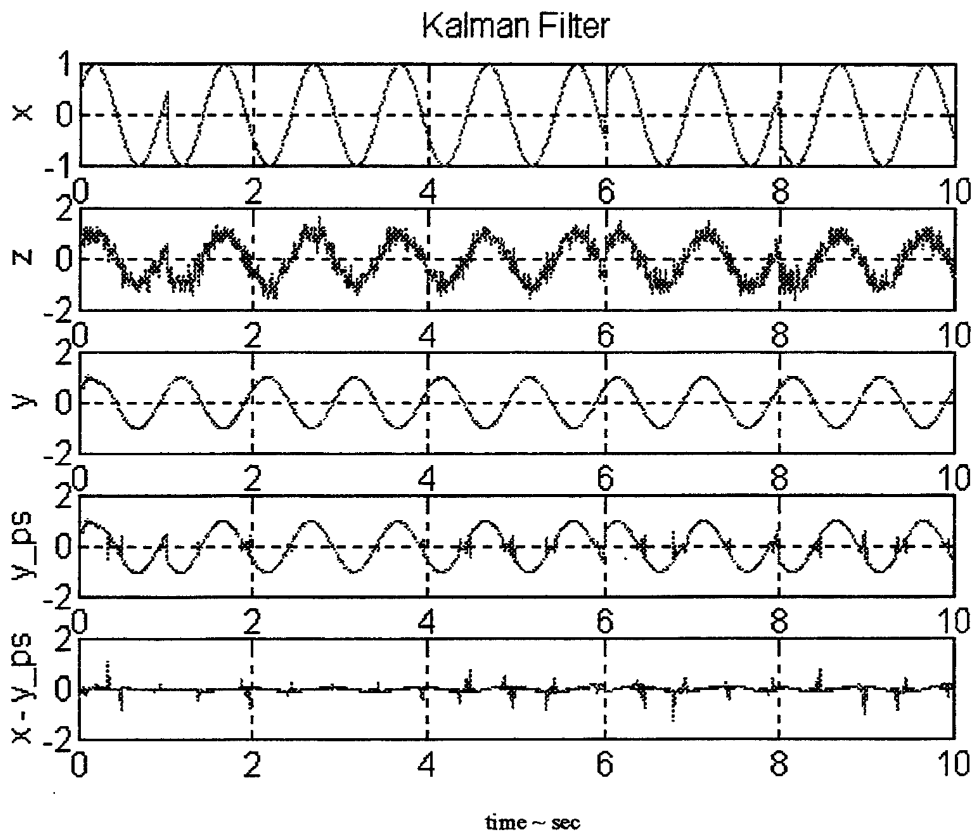


Figure 15   Kalman Filter for BPSK with Noise

C. QPSK

Next the results for a QPSK simulation are shown. These plots were generated using *kalq_sim.m*

1. Without Noise

Figure 16 shows the performance of the algorithm when there is no noise present. The true signal, x, and the estimate y_ps and the error in the estimate, x-y_ps are all plotted.

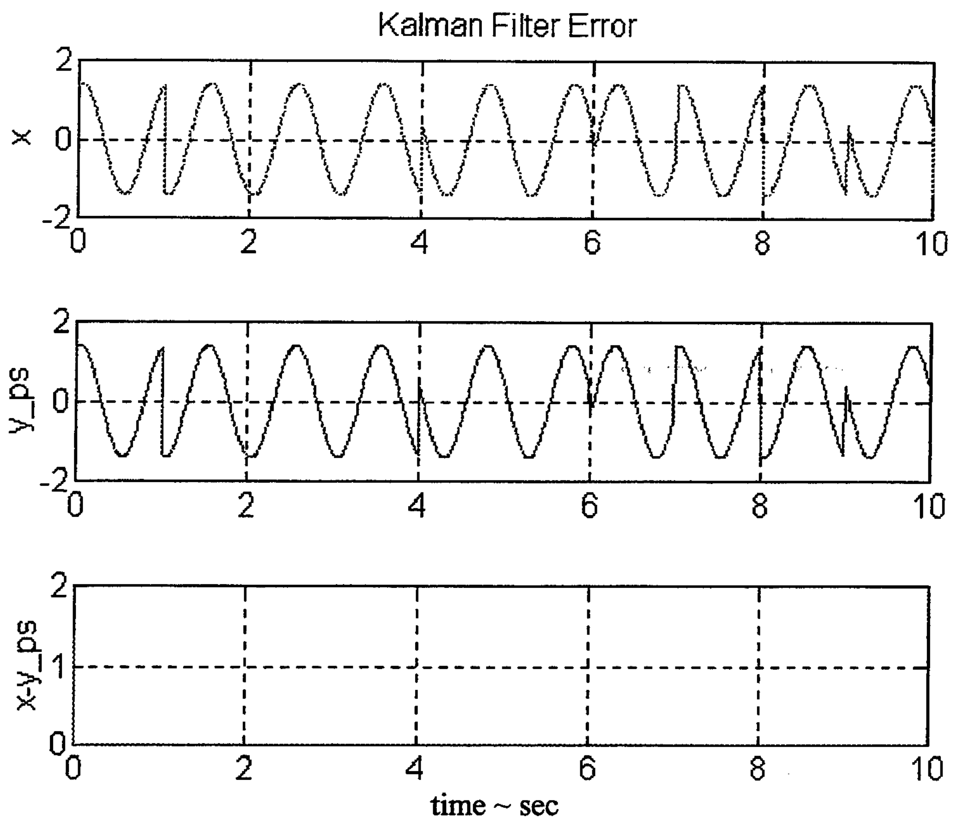


Figure 16 Kalman Filter for QPSK, Without Noise

2. With Noise

Figure 17 shows the estimation when white noise[7] is added with an SNR value of 12 dB. Figure 18 shows the error in the estimation. The carrier is still recovered, however small errors are now introduced in the estimate of the phase shifted signal. The cause of the error will be explained in the next section.

[7] Throughout this study only gaussian white noise was considered. Colored noise from other sources, such as electromagnetic interference was not considered. It was assumed that any such noise would have the same affect at both receivers and would not change the difference in the measured phase values.

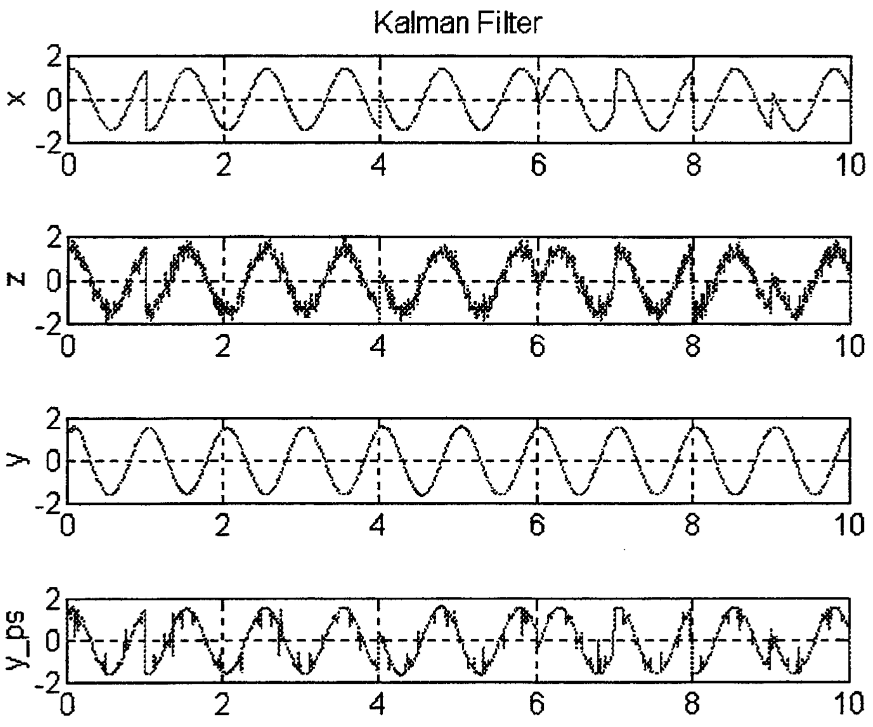


Figure 17   Kalman Filter for QPSK with Noise

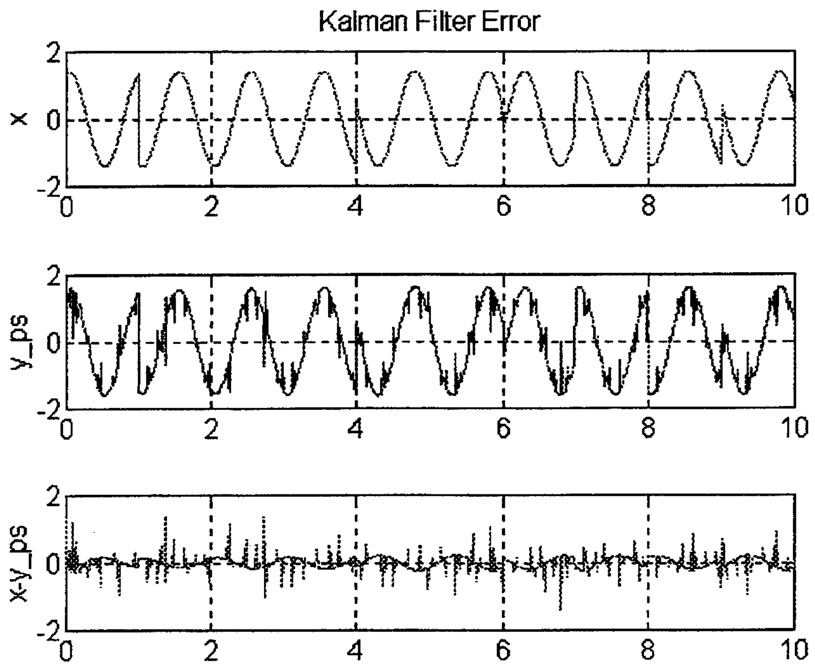


Figure 18   Error in QPSK Estimator

D. SOURCES OF ERROR IN NOISY CASE

The error in the estimation caused by the noise can best be explained by examining the BPSK simulation. In the algorithm, the Kalman filter tracks the carrier signal and estimates which generated phase shifted version has the minimum error when compared with the measured signal z. Figure 19 is a plot of the original BPSK signal, x, the signal with noise added, z; the estimated signal, y_ps; the two possible signals, x0 and x2, that must be decided between; and the error, e0 and e2, used as the basis for the decision.

The error in the signal is best described as "clicks" which occur when the Kalman filter is unable to distinguish between the correct and incorrect phase because they both have approximately the same value. The reader is invited to examine this effect between times 1 and 2 seconds. The correct estimated signal is X2 which results in the minimum error over the bit period, however at approximately time 1.9 the out of phase signal estimate X0 has the same value as X2 and the error e0 is less than e1 due to the added noise. This produces an erroneous decision to switch phase. Typically the erroneous decisions all occur when both the correct and incorrect signal estimates are near a zero crossing. This is due to the nature of the cosine function which results in cos(90°)=cos(270°)=0.

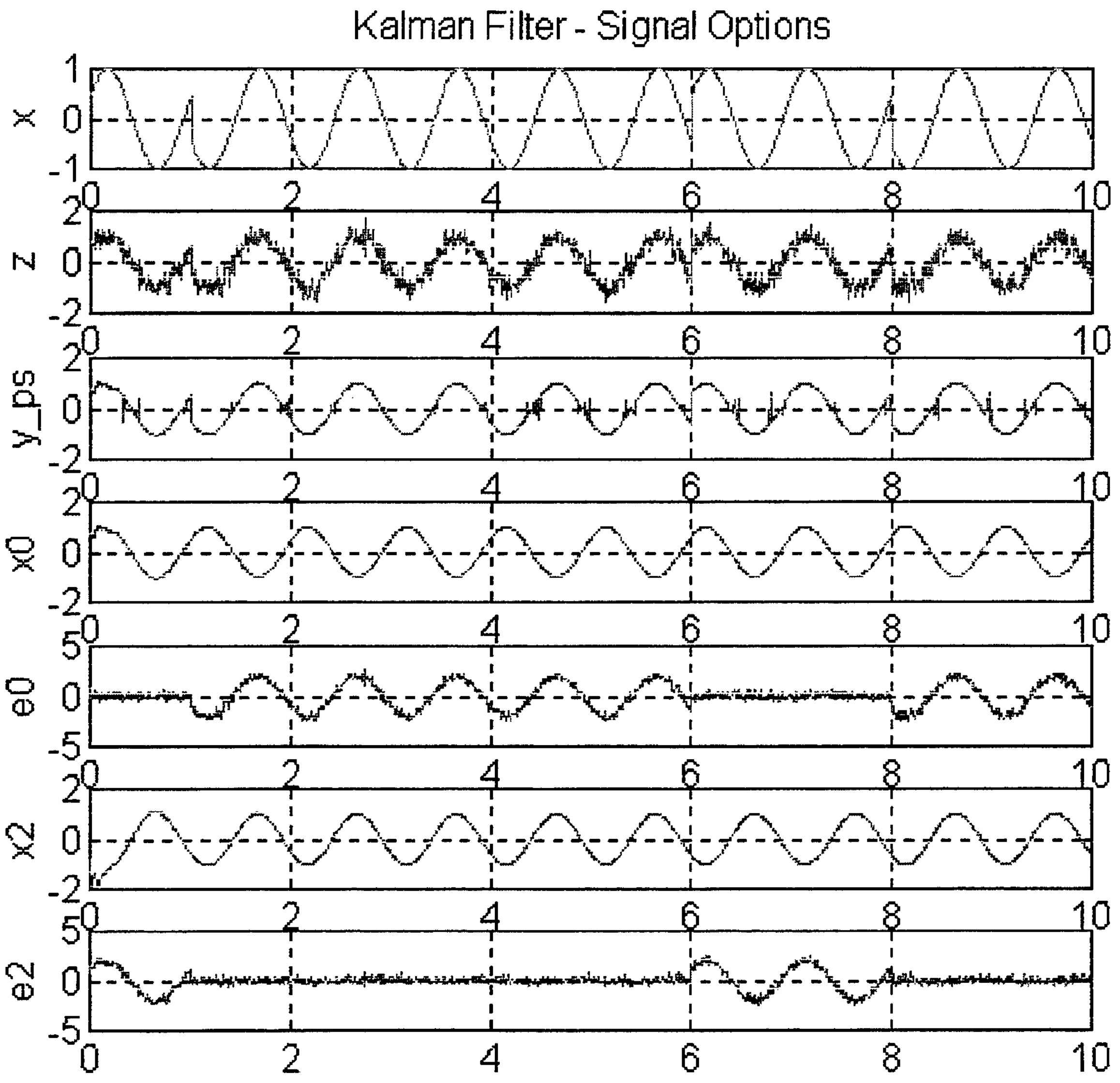


Figure 19 Kalman Filter Error Terms and Decision Options

E. UNBALANCED QPSK

As discussed in Chapter II, the GPS L1 signal is actually an unbalanced QPSK signal with differing energy in the quadrature and in-phase components. As shown in Chapter II Section F on page 15 the L1 signal is given by

$$L_1(\omega_1 t) = A_1\left[P(t)\otimes D(t)\right]\cos(\omega_1 t)+\sqrt{2}A_1\left[G(t)\otimes D(t)\right]\sin(\omega_1 t)$$

An example of the unbalanced QPSK signal is shown in Figure 20. The top graph shows the composite of the binary data representing $\left[G(t)\otimes D(t)\right]$ with the quadrature component, the middle plot shows the binary data representing $\left[P(t)\otimes D(t)\right]$ with the in-phase component. In both cases the A_1 has been normalized to 1. The last plot shows the sum of the two signals which is the resulting unbalanced QPSK signal.

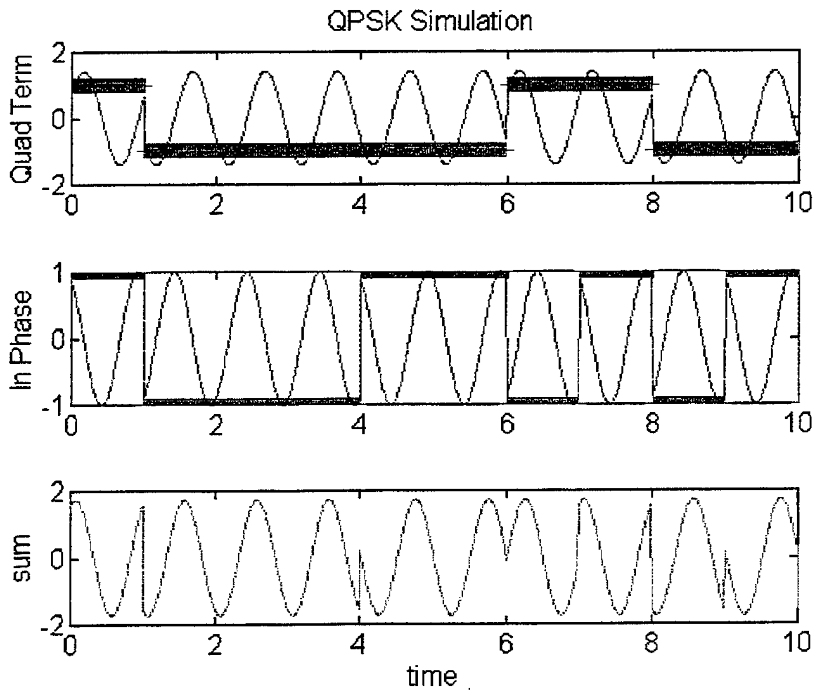


Figure 20 Unbalanced QPSK Signal Generation

When the Kalman filter is used to track this signal as shown in Figure 21, errors occur when the in-phase and quadrature terms have differing data. The filter locks on to the initial amplitude and assumes the amplitude will not change. In this example the absolute value of the amplitude is the same when the data bits are the same (both +1 or both -1) and is different with the data bit have opposite values. The state estimation was developed under the assumption the signal would have a constant amplitude. In order to resolve this problem the logic for phase change detection would need to be modified.

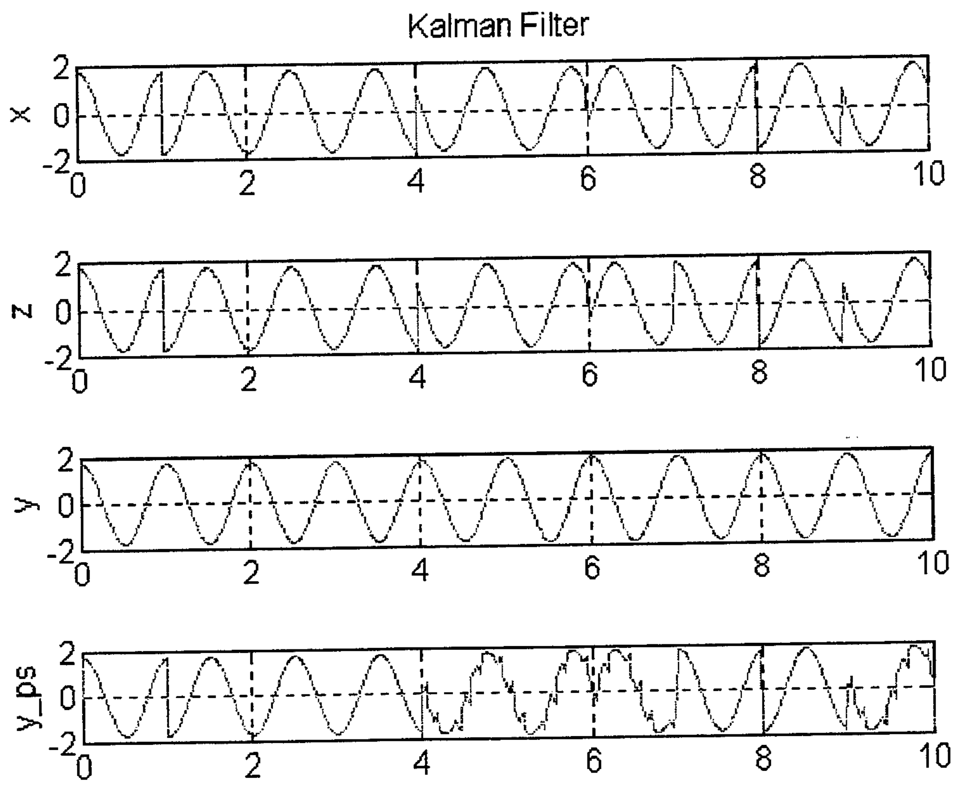


Figure 21 UQPSK Without Noise

When noise is added to the unbalanced QPSK as shown in Figure 22 the performance is the same as in the QPSK example when the data bit for the in-phase and quadrature phase term is the same, however when they are different, the performance degrades. The error in the estimation is shown in Figure 23.

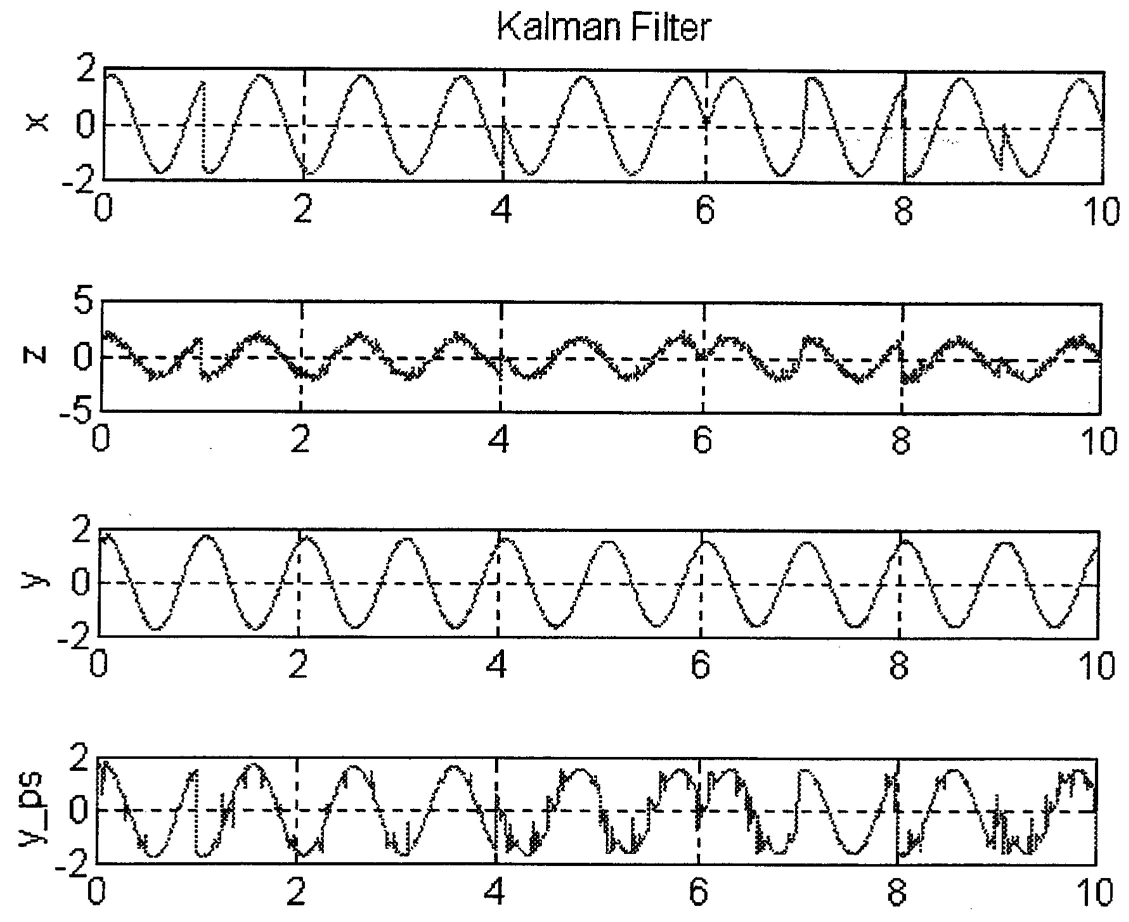


Figure 22 UQPSK with Noise

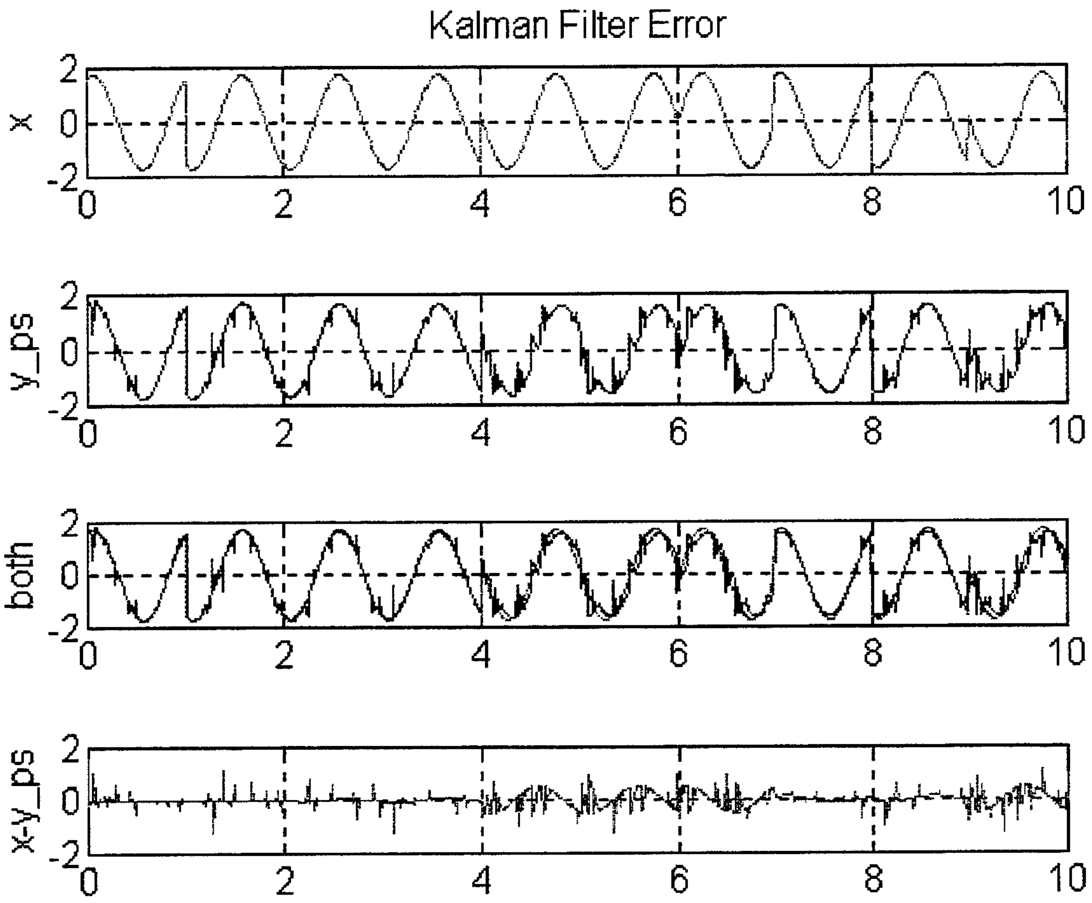


Figure 23 Error in UQPSK Estimator

F. MULTIPATH RESULTS

1. Without *A Priori* Knowledge

The multipath parameters used to generate Figure 10 are used to generate the measured signal z and the results obtained are shown below in Figure 24. The estimate, y_ps, and the direct path signal, x, are shown plotted together and the error, or the difference between x and y_ps is also shown.

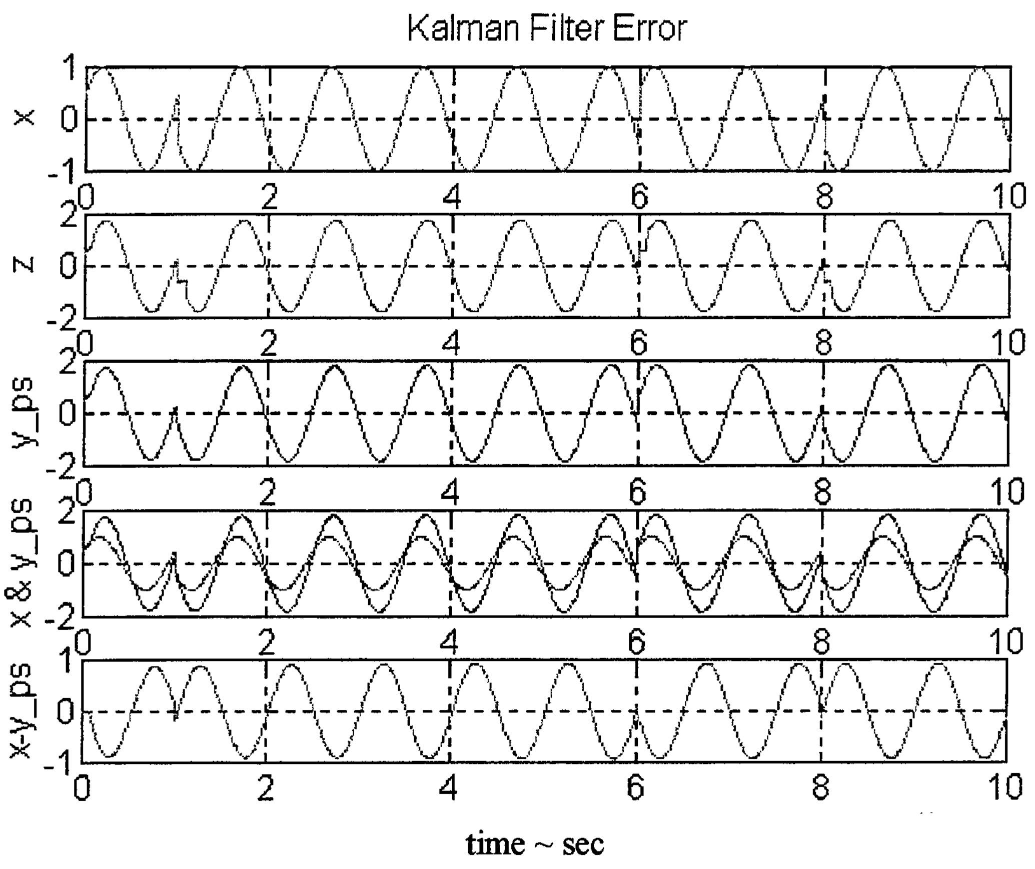


Figure 24 Results of Kalman Filter Estimation in Presence of Multipath

The Kalman filter has correctly determined the phase changes, however due to the multipath, the amplitude of the estimation is now different from the original signal. Using the larger multipath spread parameters used to generate Figure 11 the resulting phase error is observed as shown Figure 25.

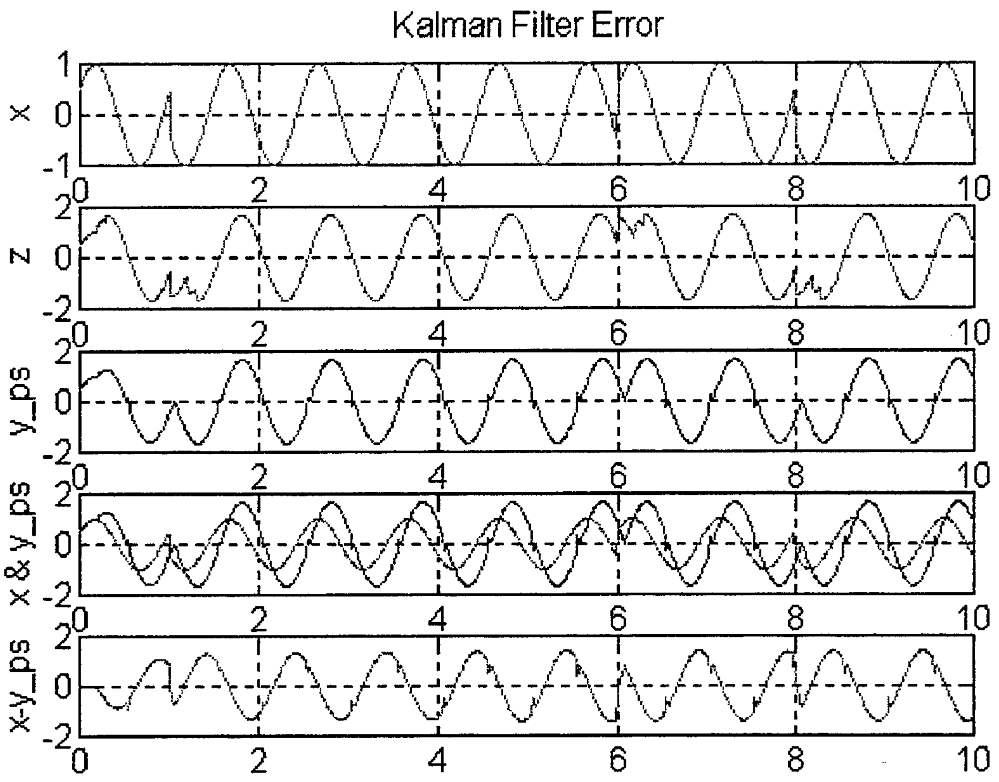


Figure 25   Results of Kalman Filter Estimation in Presence of Multipath with Increased Delay

The results of these last two plots indicate the Kalman filter performance in the presence of multipath is much worse than in the presence of white noise. If multipath is present it will introduce a phase error that will be unique to each antenna. When the differential phase is calculated it will now include the multipath phase error and will no longer be just a function of the geometry.

2. With *A Priori* Knowledge

If the multipath amplitude and time delay are known, the Kalman filter can be modified to track both the the direct path signal and delayed multipath signals. A simple case with a single multipath channel is shown below with the following assumptions:

The combined signal is given by:

$$s_m(t) = s(t) + \alpha \cdot s(t-\tau)$$

The incoming signal is not changing phase and is given by

$$s(t) = A\cos(\omega t + \theta(t))$$

where θ(t)= $a(t)\frac{\pi}{2}$ and a(t) = 0,1,2,or 3 representing the intelligence for the BPSK or QPSK modulation.

The time delay τ is converted into the phase delay ϕ

$$s(t-\tau) = A\cos(\omega(t-\tau)+\theta(t-\tau)) = A\cos(\omega t - \omega\tau + \theta(t-\tau)) = A\cos(\omega t + \phi)$$

and the composite multipath signal is now given by

$$s_m(t) = A\cos(\omega t + \theta) + \alpha\, A\cos(\omega t + \phi)$$

The plant equations are now

$$\begin{aligned} x_1 &= A\cos(\omega t + \theta) \\ x_2 &= \dot{x}_1 = -\omega\, A\sin(\omega t + \theta) \\ x_3 &= \alpha\, A\cos(\omega t + \phi) \\ x_4 &= \dot{x}_3 = -\omega\,\alpha\, A\sin(\omega t + \phi) \end{aligned}$$

$$\mathbf{x}(t) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} A\cos(\omega t + \theta) \\ -\omega\, A\sin(\omega t + \theta) \\ \alpha\, A\cos(\omega t + \phi) \\ -\omega\,\alpha\, A\sin(\omega t + \phi) \end{pmatrix}$$

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -\omega\, A\sin(\omega t + \theta) \\ -\omega^2 A\cos(\omega t \theta) \\ -\omega\,\alpha\, A\sin(\omega t + \phi) \\ -\omega^{\,2}\alpha\, A\cos(\omega t + \phi) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\omega^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\omega^2 & 0 \end{pmatrix} \cdot \begin{pmatrix} A\cos(\omega t + \theta) \\ -\omega A\sin(\omega t + \theta) \\ \alpha\, A\cos(\omega t + \phi) \\ -\omega\,\alpha\, A\sin(\omega t + \phi) \end{pmatrix} = \mathbf{A}\mathbf{x}(t)$$

Since *a priori* knowledge is known the values for the estimate at time zero can now be initialized as follows:

$$\hat{x}(0|-1) = \begin{bmatrix} A \\ 0 \\ \alpha A \cos(\phi) \\ -\omega\alpha A \sin(\phi) \end{bmatrix}$$

Figure 26 shows the line of sight signal, delayed signal, and the measured signal z which were generated using α=0.5 and τ=10 with Ts=.1. For this example the multipath environment is assumed to be static. In reality, the multipath environment would be much more dynamic due to the satellites orbit, the satellites changing attitude, and the changing position of the GPS satellites.

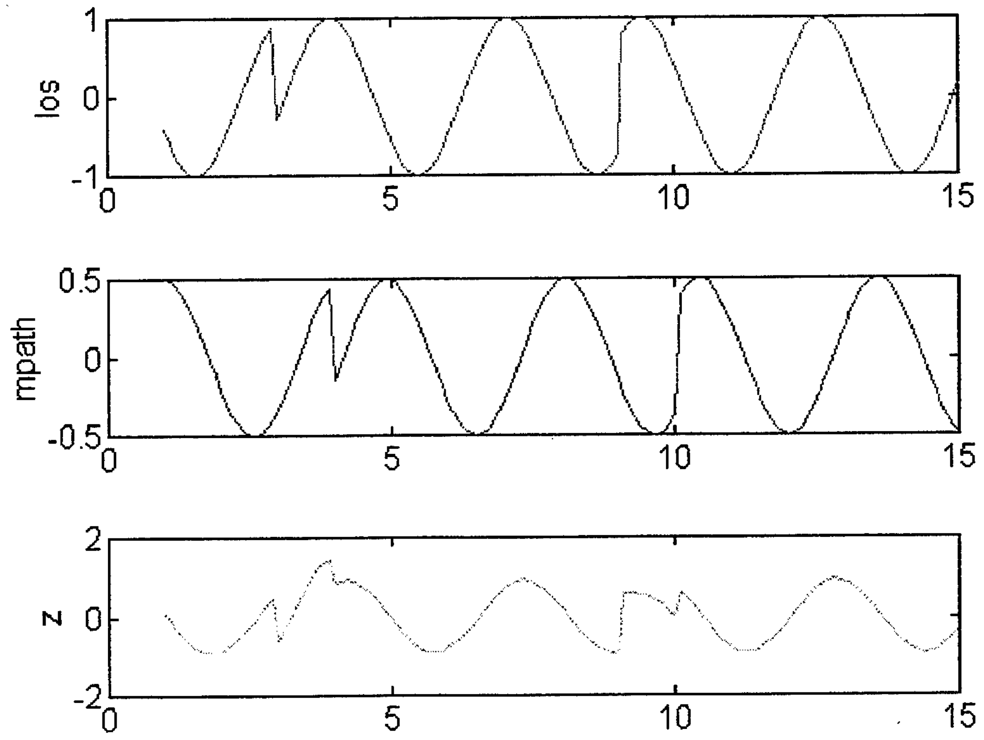


Figure 26 Example Signal for A Priori Case

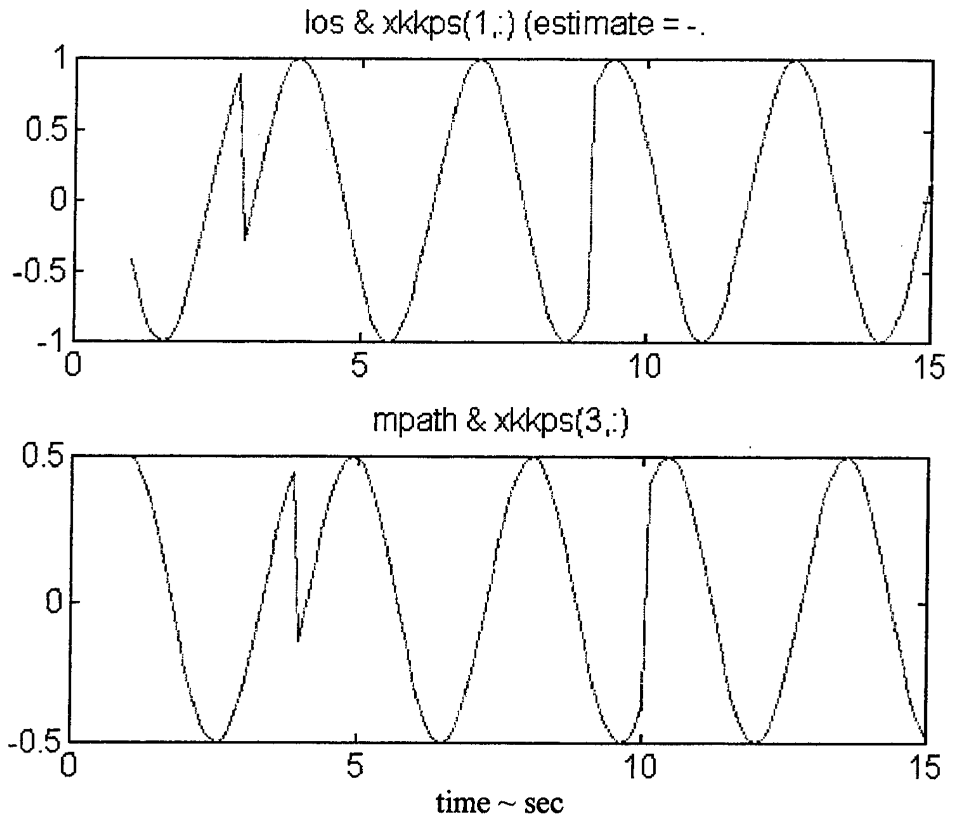


Figure 27 A Priori Knowledge Results Showing Signals and Estimates

Figure 27 shows the estimations and original signals plotted on top of each other. This demonstrates how, given *a priori* knowledge, the line of sight and delayed signal can be tracked. Now there is no phase error due to the multipath signal, unlike the examples shown without *a priori* knowledge when the filter tracked the phase of the combined signal.

When noise is added in this simple case with 12 dB SNR as shown in Figure 28 and Figure 29 there is also improved performance with *a priori* knowledge.

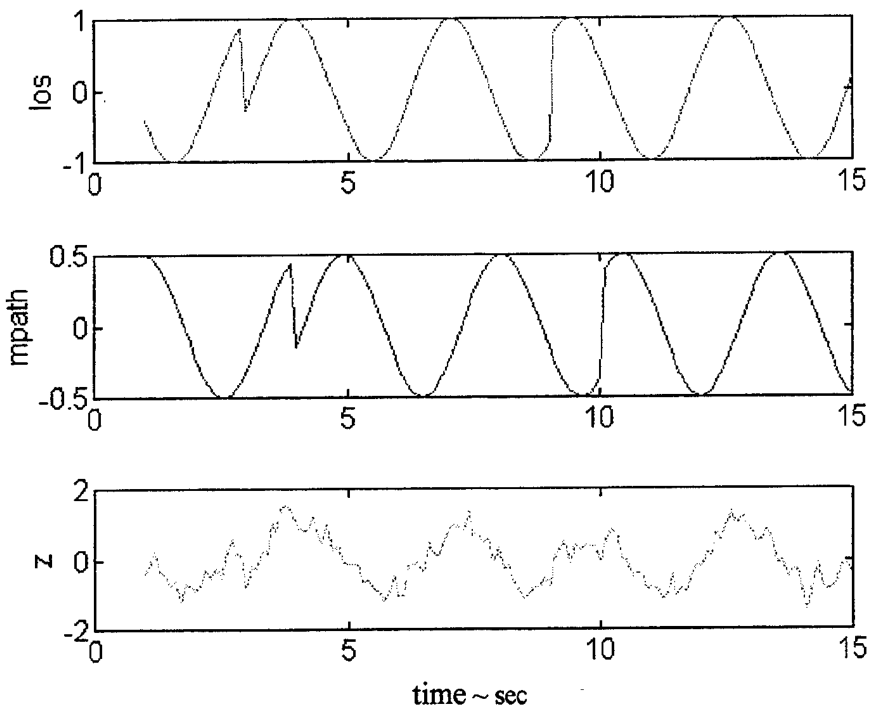


Figure 28 Generated Signal for A Priori Example with Noise

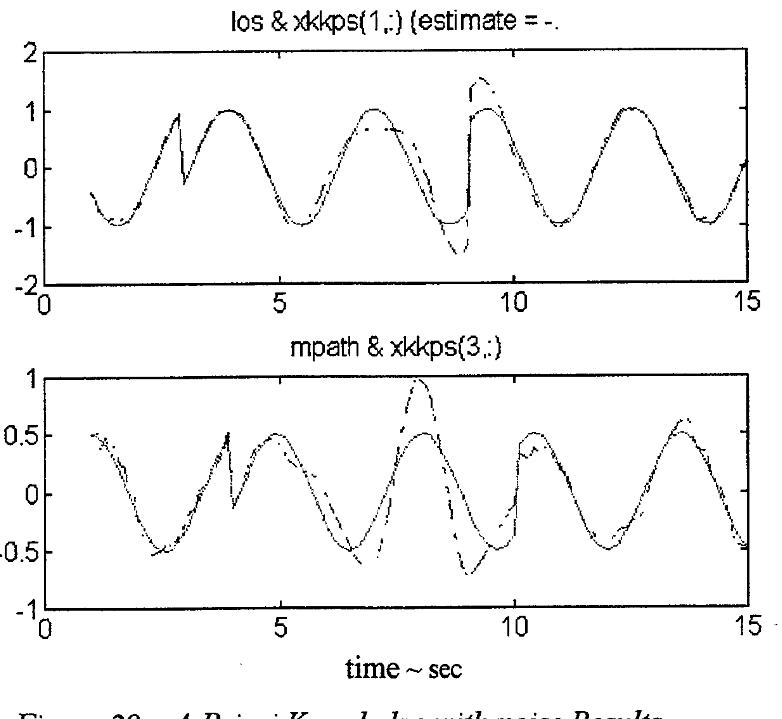


Figure 29 A Priori Knowledge with noise Results

G. EFFECT OF Q MATRIX ON KALMAN GAINS

The random process forcing input covariance matrix Q is a measure of the expected excitation of the noise. It is used in the calculation of the Kalman gains as shown in Chapter IV, Section A-3 on page 26 which are restated here:

$$G(k) = P(k|k-1) \cdot C^T \cdot \left[C \cdot P(k|k-1) \cdot C^T + R\right]^{-1}$$

$$P(k|k) = [I - G(k) \cdot C] \cdot P(k|k-1)$$

$$P(k+1|k) = \phi \cdot P(k|k) \cdot \phi^T + Q$$

The Q matrix is a function of the state noise coefficient matrix, Γ, and the variance of the forcing function, w, as shown below.

$$\Gamma = \begin{bmatrix} \frac{dt^2}{2} \\ dt \end{bmatrix}$$

$$Q = \Gamma \cdot \Gamma^T \cdot \text{var}[w]$$

$$Q = \begin{bmatrix} \frac{dt^4}{4} & \frac{dt^3}{2} \\ \frac{dt^3}{2} & dt^2 \end{bmatrix} \cdot \text{var}[w]$$

As there is no forcing function, w, the value of Q was initially set to zero, however in order to determine its impact, different values for the variance of w, var[w], were used. The filter results and Kalman gain values G1 and G2 are shown in Figure 30 through Figure 33 for different values of var[w] and Q. As the value for var[w] is increased, the steady state gain increases. The only noticeable effect occurs when var[w] is greater than 100. With an increase in var[w] and Q, there is also an increase in the Kalman gains. This results in more responsiveness. If the gain values are too large the filter will be too responsive resulting in more errors.

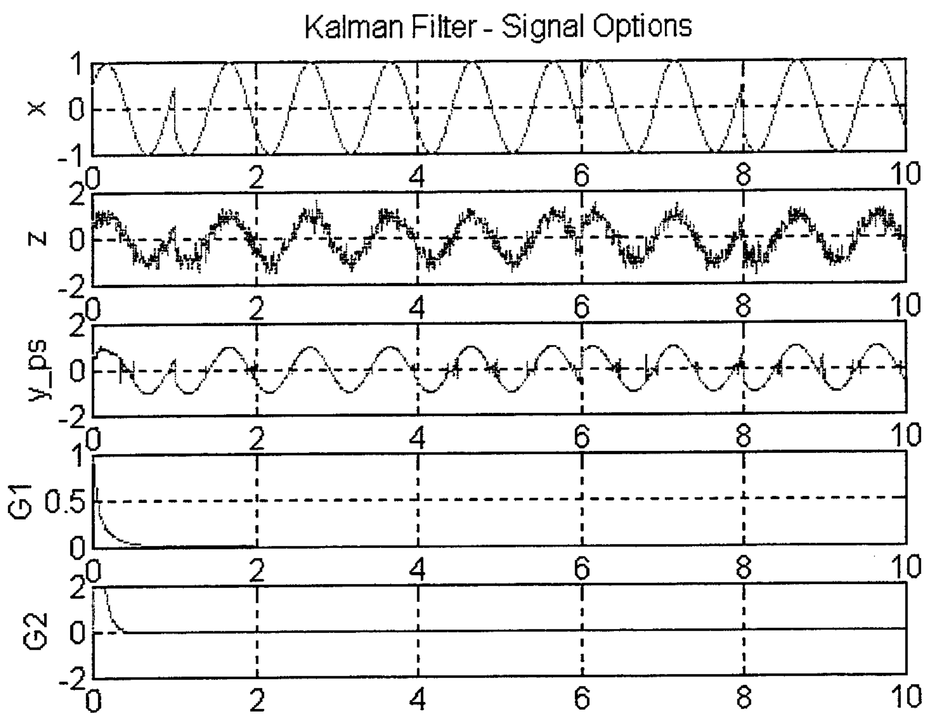


Figure 30 Effect of Gain Values with var[w]=0, $Q=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$

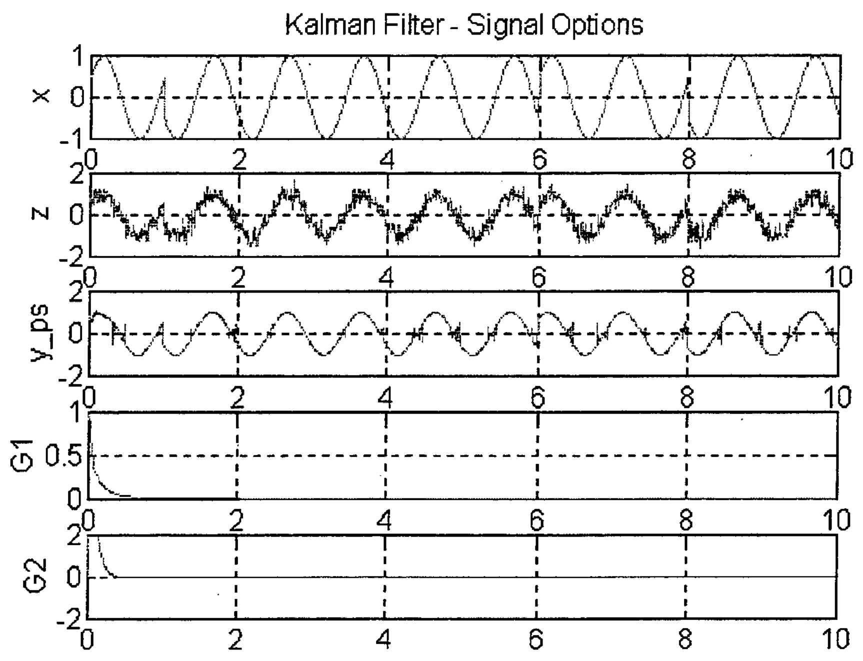


Figure 31 Effect of Gain Values with var[w]=10, $Q = \begin{bmatrix} 0.0003 & 0.005 \\ 0.005 & 0.1 \end{bmatrix}$

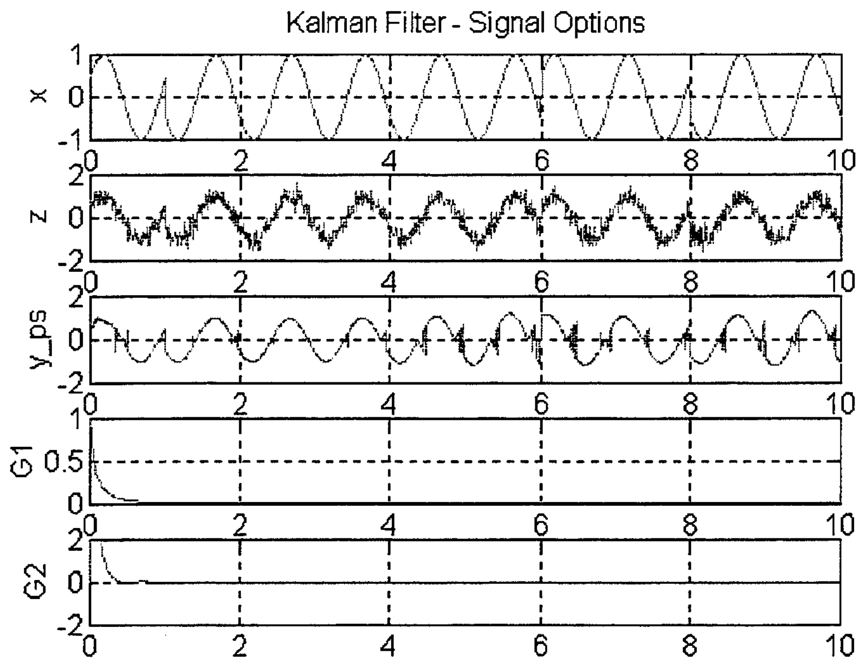


Figure 32 Effect of Gain Values with var[w]=100, $Q = \begin{bmatrix} 0.0025 & 0.05 \\ 0.05 & 1 \end{bmatrix}$

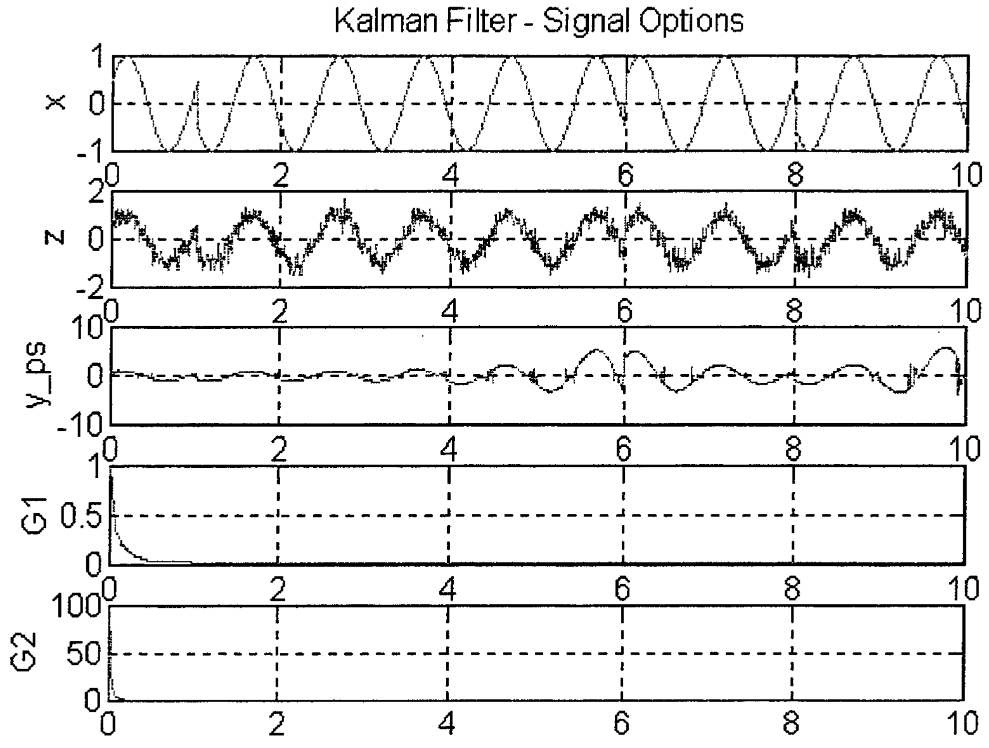


Figure 33 Effect of Gain Values with var[w]=150, $Q = \begin{bmatrix} 0.0038 & 0.075 \\ 0.075 & 1.5 \end{bmatrix}$

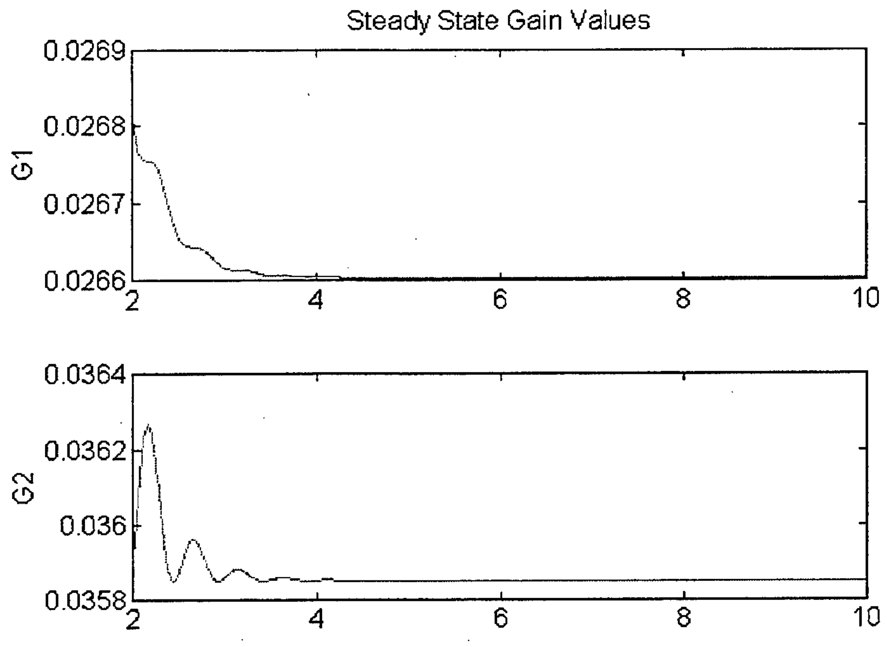


Figure 34 Steady State Kalman Gains with var[w]=150

The steady state gain values after time t=2 seconds are shown in Figure 34. For all other values of var[w] shown, the corresponding steady state Gain values approached zero.

H. SUMMARY

This chapter has presented the results of using the Kalman filter to estimate the phase in BPSK, QPSK, and Unbalanced QPSK signals to simulate the various formats of a GPS signal. Results have been shown for the performance with and without Gaussian white noise and with multipath interference.

The designed filter works best with the BPSK signal. Gaussian white noise degrades performance due to the effects of the correct and incorrect phase having the same value at a zero crossing. It may be possible to modify the phase switching logic to reduce this effect by allowing a phase change to only occur at or near the expected time of the end of a bit period or when the error differences are greater than some minimum value.

The multipath interference was shown to result in errors in the estimate's phase and amplitude. This error is caused by the filter tracking the phase of the combined multipath signal. It was shown that with *a priori* knowledge of multipath delay and amplitude, the filter is able to discriminate between the direct path and the delayed signals and to eliminate this error. Sources of the *a priori* information might include ground testing by placing a receiver into a self-survey mode and allowing the GPS satellites to pass overhead. (Lightsey) It may also be possible to use the technique suggested by Kumar and Lau to estimate the multipath parameters through seismic deconvolution

For this case it was assumed that the multipath environment was not changing. In reality the multipath environment would be very dynamic as a result of the satellites orbit and changing attitude. Additionally, the attitude determination and control system only uses 4 out of possibly 11 visible satellites. The satellites selected would also effect the multipath environment. To overcome this may require frequent re-initialization of the Kalman filter.

VI. CONCLUSION

A. SUMMARY

This study has focused on the carrier phase signal which is used by GPS based attitude determination systems. A GPS carrier signal and a multipath environment were described and modeled. A Kalman filter was developed to track the GPS signal and results were shown for Gaussian noise and in the presence of multipath.

The developed Kalman filter was shown to work best for the L2, BPSK signal due to the minimum complexity involved in the phase changing logic. It was shown that the filter is capable of discriminating between the line of sight and delayed signals and performed well in the presence of multipath, given good *a priori* knowledge.

More refinement is required in the phase change detection logic to eliminate errors caused by the error terms having the same value at a zero crossing. The Q excitation matrix needs considerable adjustment to handle changing phase in the presence of multipath. The source of the *a priori* information needs to be developed.

B. OTHER METHODS OF MULTIPATH MITIGATION FOR POSSIBLE STUDY

As multipath accounts for 90% of the error in an attitude determination system, much work is currently being done in this area. Three categories of multipath mitigation methods that have been used include: 1) Methods involving the geometry of the multipath, 2) Analyzing the consistence of the different code measurements and 3) Exploiting information coming from different channels.

Of these three, the easiest to implement involves geometry. One approach is to focus on improving the antenna or its position. This may be accomplished by placing the antenna in isolated locations minimizing the multipath, something that is not always possible on a satellite. Another method of multipath rejection is achieved by reducing the left-hand circularly polarized gain (LHCP) of the antenna without reducing the right-hand circularly polarized gain (RHCP). This is because the satellite signals are RHCP and the reflected multipath signals tend to be either linearly polarized or even LHCP, depending on the reflecting surface. (Van Dierendonck) An example of an antenna designed specifically to reduce multipath effects is the choke ring antenna. This antenna is comprised of vertical-aligned concentric rings centered about the antenna element that are connected to the ground plane. The vertical rings shape the antenna pattern such that the multipath signals incident on the antenna at the horizon and negative elevation angles are attenuated (Kaplan).

Other methods of dealing with the multipath problem involve modification to the GPS receiver for code measurements. One approach involves reducing the early-late delay spacing among the correlators in the GPS receiver code lock loop. This technique provides limited results if the early-late spacing is smaller than the initial delay error due to multipath. (Kumar)

An example of the third category is the method presented by Kumar and Lau involving the use of optimal deconvolution to estimate the impulse response of the effective multipath channel and obtain an inverse filter which equalizes the multipath channel (Kumar) Another technique involves adaptively estimating the spectral parameters (frequency, amplitude, phase offset) of multipath in the associated SNR and then constructing a profile of the multipath error in the carrier phase. A multipath correction is then made by subtracting the profile from the actual phase measurement. (Comp)

LIST OF REFERENCES

Comp, C. J, and Axelrad P., "An Adaptive SNR-Based carrier Phase Multipath Mitigation Technique," *Institute of Navigation, Proceedings of the ION GPS-96 Conference*, 1996.

Cohen, C. E., "Attitude Determination," *Progress in Astronautics and Aeronautics* Vol. 164, pp. 519-537, 1996.

ICD-GPS-2000 NAVSTAR *GPS Space Segment/Navigation User Interfaces* Rockwell International Corporation, 1987.

Kaplan, E. D., *Understanding GPS Principles and Applications*, Artech House Publishers, Norwood, MA, 1996.

Kirk, D. E., "Optimal Estimation: an Introduction to the Theory and Applications" unpublished lecture notes, Naval Postgraduate School, Monterey, CA, 1975.

Kumar, R. and Lau, K., "Deconvolution Approach to Carrier and Code Multipath Error Elimination in High Precision GPS," *Institute of Navigation, Proceedings of 1996 National Technical Meeting,* 1996.

Larson, W. J., and Wertz, J. R., *Space Mission analysis and Design,* 2nd ed., Microcosm, Inc., Torrance, CA, 1992.

Lightesy, E. G., "Spacecraft Attitude Control Using GPS Carrier Phase," *Progress in Astronautics and Aeronautics* Vol. 164, pp. 461-480, 1996.

Struze, M. A., "Digital Signal Processing Techniques for GPS Receivers," *NTC '83 IEEE National Telesystems Conference,* 1983.

Van Dierendonck, A. J., "GPS Receivers," *Progress in Astronautics and Aeronautics* Vol. 163, p. 332.

APPENDIX A MATLAB PROGRAMS

A. DO_KAL.M

```
% File Name: do_kal.m
% Last Updated: 18 Apr 97
% Written By: T. H. Newman
% Operating Environment: Windows 95
% Matlab Version 4.2c.1
%
% Description: Script file to set up and run kalman filter simulations

clear

ui_vars

% End of File do_kal.m
```

B. G1.M

```
% File Name: g1.m
% Last Updated: 18 Oct 96
% Written By: T. H. Newman
% Operating Environment: Windows 95
% Matlab Version 4.2c.1
%
% Description: This function generates the G1(t) signal
% for GPS C/A Code generator
%
%

function [new_R, g] = g1(R, x1)

if x1==1
  R = ones(size(R)); % Reset to all ones at X1 epoch
end;
new_R=gen_poly([1,3,10],R);

g=R(10);



% End of File g1.m
```

C. G2.M

```
% File Name:  g2.m
% Last Updated:  18 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  This function generates the G2(t) signal
% for GPS C/A Code generator
%
%

function [new_R, g] = g2(R, x1, sv)

if x1==1
  R = ones(size(R));  % Reset to all ones at X1 epoch
end;
new_R = gen_poly([1 2 3 6 8 9 10], R);

%new_R(1) = mod2(R(2)+R(3)+R(6)+R(8)+R(9)+R(10));
%new_R(2:10) = R(1:9);


  % phase select logic, see p. 92 KAPLAN
  % This changes based on satellite choosen.
sv_tap = [  2 6
        3 7
        4 8
        5 9
        1 9
        2 10
        1 8
        2 9
        3 10
        2 3
        3 4
        5 6
        6 7
        7 8
        8 9; 9 10;1 4;2 5;3 6; 4 7; 5 8; 6 9; 1 3; 4 6; 5 7; 6 8; 7 9;
        8 10; 1 6; 2 7; 3 8; 4 9; 5 10; 4 10; 1 7; 2 8; 4 10];


g=xor( R(sv_tap(sv,1) ), R(sv_tap(sv,2)) );

% End of File g2.m
```

D. GEN_POLY.M

```
% File Name:  gen_poly.m
% Last Updated:  18 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description: This function generates the linear code generators
% the polynomial is of the form 1 + sum(x(i)) where
% sum(x(i)) means that the output of the i'th cell of the register
% is used as the input to the modulo-2 adder (exclusive or)
% and the 1 means that the output of the adder is fed to the first cell
%
% so 1 + x1 + x3 + x10 could be generated with poly =[1 1 3 10] and R
% is the register to perform the operation on

function r = gen_poly(poly, R)

result = mod2( sum(R(poly(2:length(poly) ) )));        % xor registers according to
                                                 %    polynomial

r(2:length(R)) = R(1:(length(R)-1));            % shift registers right
r(poly(1))=result;                                    % put result in register according
                                                                                  %    to polynomial

% End of File gen_poly.m
```

E. INIT.M

```
% File Name:  init.m
% Last Updated:  11 Nov 96
% Written By:  T. H. Newman
% Operating Environment:  Sun Openwin & Win 95
% Matlab Version 4.2c.1
%
% Description:  This script file sets the parameters for
% the GPS signal simulation

global w;

% my_glb;
```

```
fo = 10.22999999543e6;
f1 = 120*fo;                % frequency of L2 signal
f2 = 154*fo;                % frequency of L1 signal

To = .001;

code_length = 10;
[s1,s2,s3]=sig_gen(code_length);
sig1=[(1:length(s1))',s1'];
sig2=[(1:length(s2))',s2'];
sig3=[(1:length(s3))',s3'];

kal_init;

% End of File init.m
```

F. KAL_INIT.M

```
% File Name: kal_init.m
% Last Updated:  27 Feb 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Initialization file for Kalman Filter
%


global Pkkm1;
global R;
global Q;
global I;
global Phi;
global xkkm1;
global Tb;
global Noise_gain;
global dt;

global w;

% my_glb;

A=[0 1; -w^2 0]; % x1 = cos(wt)
                                   % x2 = x1_dot = -w*sin(wt)
                                   % x2_dot = -w^2*cos(wt)
                                   % x = [x1 x2]'
```

```
                                        % x_dot = A*[x1_dot, x2_dot]
                                        %       => A = [0 1; -w^2 0]'

B=[0 w^2]';                             % x_dot = Ax + Bu
                                        %    u is control input


C=[1 0];                          % will use to get x1 estimate

D=0;

I=eye(2);                               % Identity Matrix

[Phi,Del]=c2d(A,B,dt);                  % converts the continuous-time system:
                                        % x_dot = Ax + Bu
                                        % to the discrete-time state-space
                                        % system:
                                        %       x[n+1] = Phi * x[n] + Del * u[n]
                                        % assuming a zero-order hold on the
                                        % inputs and sample time dt


Pkkm1=1e6*I;                  % Initialize P(k|k-1)  = M
                                        % where M = covariance of initial state
                                        %       want this number large to give
                                        %       weight to 1st measurment
R=1;                                    % var[e(0|0)] = var[v(0)] = R
                                        % v is reference input
                                        %   (noise in measurement)

%Q=0*I;                                 % Q is measure of exitation by forcing function

var_w=150;
Q = [ (dt^4)/4 (dt^3)/2; (dt^3)/2 dt^2 ]*var_w;

xkk = [0 0; 0 0];  % initialize x(k|k) to all zeros
                                        %       x(k|k) is state of plant at time k given k observations
xkkps = [0 0; 0 0];

xkkm1 = [0,0; 0,0];                     % initialize x(k|k-1)  => x(0|-1) = x_o = 0


% -------------------------
% clear all the variables used for saving plotting information
clear my_rand;
clear x;
clear sm;
clear x_delay;
clear z;
clear y;
clear xkk_ps;
clear t;
```

```
% End of File kal_init.m
```

G. KALMAN.M

```
% File Name:  kalman.m
% Last Updated:  28 Feb 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Kalman filter for tracking phase of a
%  QPSK signal
%
% Based on Professor Titus' algorithim which follows Prof. Kirk's
% NPS lecture notes " Optimal Estimation: An Introduction to the
% Theory and Applications" 1975  All equation numbers are referenced
% to the lecture notes.

function  y=kalman(z)

global Pkkm1;   % Values initialized in kal_init
global R;
global Q;
global I;
global Phi;
global xkkm1;
global w;

C=[1,0];
Cps = [0,1];

G=Pkkm1*C'*inv(C*Pkkm1*C' +R);                    % Calculate Gain at time k
                                                                  %        G(k) = P(k|k-
1)*C'*inv(C*P(k|k-1)*C' + R)
                                                                  %        eqn 4.3-102a

Pkk=(I-G*C)*Pkkm1;      % Calculate P(k|k)
                                          %        P(k|k) = [I -G(k)*C(k)]*P(k|k-1)  eqn 4.3-103a

Pkkm1=Phi*Pkk*Phi' + Q;            % P(k|k-1)

e0 = z-C*xkkm1;                          x0=xkkm1+G*e0;
e1 = z-Cps*xkkm1/w;     x1=Cps*(xkkm1+G*e1)/w;
e2 = z+C*xkkm1;                                       x2=-(xkkm1+G*e2);
e3 = z+Cps*xkkm1/w;                   x3= -Cps*(xkkm1+G*e3)/w;

if abs(e0)<=abs(e1);
```

```
  e=e0;                % Smallest error is with
  xkk = xkkm1 +G*(e);    %   old phase
  xkkps = xkk;
else;
  e=e1;                    % Smallest error is +90
  xkk = xkkm1 +G*(e);  % is
  xkkps = Cps*xkk/w;
end

if abs(e2) <= abs(e)
  e=e2;
  xkk = xkkm1 + G*(e);
  xkkps = -xkk;              % Smallest error is -90
end;

if abs(e3) <= abs(e)
  e=e3;
  xkk = xkkm1 + G*e;            % Smallest error is 180
  xkkps = -Cps*xkk/w;
end;

xkkm1=Phi*xkk;                  % eqn 4.3-89a

y1= C*xkk;                      % report the non shifted version

y=[y1(1,1);  xkkps(1); x0(1); x1(1); x2(1); x3(1)];


% End of File kalman.m
```

H. KALMAN_U.M

```
% File Name:  kalman_u.m
% Last Updated:  26 May 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Kalman filter for tracking phase of a
%  unbalanced QPSK signal
%
% Based on Professor Titus' algorithim which follows Prof. Kirk's
% NPS lecture notes " Optimal Estimation: An Introduction to the
% Theory and Applications" 1975  All equation numbers are referenced
% to the lecture notes.
```

```
function  y=kalman(z)

global Pkkm1;   % Values initialized in kal_init
global R;
global Q;
global I;
global Phi;
global xkkm1;
global w;
global A1;
global A2;

C=[1,0];
Cps = [0,1];

G=Pkkm1*C'*inv(C*Pkkm1*C' +R);                    % Calculate Gain at time k
                                                              %        G(k) = P(k|k-
1)*C'*inv(C*P(k|k-1)*C' + R)
                                                              %        eqn 4.3-102a

Pkk=(I-G*C)*Pkkm1;      % Calculate P(k|k)
                                    %        P(k|k) = [I -G(k)*C(k)]*P(k|k-1)  eqn 4.3-103a

Pkkm1=Phi*Pkk*Phi' + Q;          % P(k|k-1)

F=(1/A2);                              % Factor for unbalanced quadature phase term
cos_wt=C*xkkm1;
sin_wt=-Cps*xkkm1/w;

e0 = z-(cos_wt + sqrt(2)*sin_wt);
e1 = z-(cos_wt - sqrt(2)*sin_wt);
e2 = z+(cos_wt - sqrt(2)*sin_wt);
e3 = z+(cos_wt + sqrt(2)*sin_wt);

if abs(e0)<=abs(e1);
   e=e0;                 % Smallest error is with
   xkk = xkkm1 +G*(e);    %   old phase
   xkkps = (cos_wt+sqrt(2)*sin_wt);
else;
   e=e1;                        % Smallest error is +90
   xkk = xkkm1 +G*(e);  % is
   xkkps = ( cos_wt - sqrt(2)*sin_wt );
end

if abs(e2) <= abs(e)
   e=e2;
   xkk = xkkm1 + G*(e);
   xkkps = (cos_wt - sqrt(2)*sin_wt);              % Smallest error is -90
end;

if abs(e3) <= abs(e)
   e=e3;
   xkk = xkkm1 + G*e;          % Smallest error is 180
   xkkps  = (cos_wt + sqrt(2)*sin_wt);
end;

xkkm1=Phi*xkk;                  % eqn 4.3-89a

y1= C*xkk;                      % report the non shifted version

y=[y1(1,1);  xkkps(1)];


% End of File kalman_u.m
```

I. KALMAN2.M

```
% File Name:  kalman2.m
% Last Updated:  3 May 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Kalman filter for tracking phase of a
%  BPSK signal.
%
% Based on Professor Titus' algorithim which follows Prof. Kirk's
% NPS lecture notes " Optimal Estimation: An Introduction to the
% Theory and Applications" 1975  All equation numbers are referenced
% to the lecture notes.

function  y=kalman(z)

global Pkkm1;  % Values initialized in kal_init
global R;
global Q;
global I;
global Phi;
global xkkm1;
global w;

C=[1,0];
Cps = [0,1];

G=Pkkm1*C'*inv(C*Pkkm1*C' +R);              % Calculate Gain at time k
                                                             %        G(k) = P(k|k-
1)*C'*inv(C*P(k|k-1)*C' + R)
                                                             %        eqn 4.3-102a
```

```
Pkk=(I-G*C)*Pkkm1;      % Calculate P(k|k)
                                      %        P(k|k) = [I -G(k)*C(k)]*P(k|k-1)  eqn 4.3-103a

Pkkm1=Phi*Pkk*Phi' + Q;          % P(k|k-1)

e0 = z-C*xkkm1; x0=xkkm1+G*(e0);
e2 = z+C*xkkm1; x2=-(xkkm1+G*e2);


if abs(e0)<= abs(e2);
  e=e0;                    % Smallest error is with
  xkk = xkkm1 +G*(e);    %   old phase
  xkkps = xkk;
else
  e=e2;
  xkk = xkkm1 + G*(e);
  xkkps = -xkk;                % Smallest error is -90
end;


xkkm1=Phi*xkk;                        % eqn 4.3-89a

y1= C*xkk;                            % report the non shifted version

y=[y1(1,1);  xkkps(1)];


% End of File kalman2.m
```

J. MOD2.M

```
% File Name:  mod2.m
% Last Updated:  18 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description: Returns the modulo 2 value of input
% if input is multiple of 2 returns 0
% else returns 1
%
%

function x = mod2(y)

if (y/2)*10 == round(y/2)*10
  x = 0;
```

```
else
  x =1;
end


% End of File mod2.m
```

K. MPATH.M

```
% File Name:  mpath.m
% Last Updated:  28 Apr 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  This function simulates the multipath environment

function sm = mpath(s)

global delay
global alpha
global tau

len=length(delay);

if ( len > 1 )
        delay (2:len) = delay(1:len-1);  % Shift delay vector
        delay(1) = s;
        sm = sum(delay(tau).*alpha);
else
   delay(1)=s;
   sm=s;
end;


% End of File mpath.m
```

L. MPATH_AP.M

```
% File Name:  mpath_ap.m
% Last Updated:  4 Jun 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
```

```
% Description:  Kalman filter for multipath with a priori
% information.

clear;    % clear old variables

w=2;      % = 2*pi*f
Tf=15;  % finish time ~ sec
dt=.1;

noise_fac=0;  % change scale on noise

%multipath a priori information
alpha = .5;   % delayed signal amplitude
tau = 10;     % time of delay
Amp = 1;       % amplitude of line of sight signal


A=[0 1 0 0; -w^2 0 0 0;0 0 0 1;0 0 -w^2 0];
B=[0 w^2 0 0]';
C=[1 0 1 0];

[Phi,Del]=c2d(A,B,dt);

I=eye(4);
Pkkm1=1e6*I;
R=1;
Q=0.1*I;
kmax=Tf/dt +1 ;


switch1 = 3;    % time of phase changes
switch2 = 9;

time=0;
for (i=1:kmax)                    % generate line of sight signal
  t(i)=time;
  if t(i) <= switch1
              los(i) = cos(w*t(i));   % line of sight direct path signal
  elseif t(i) >= switch1 & t(i) <= switch2
     los(i) = sin(w*t(i));
  else
     los(i) = cos(w*t(i));
  end;
  time=time+dt;
end;
t1=t;
time = t;
clear t;

for i=1:length(los)-tau           % generate delayed signal
  x(i) = los(i+tau);                        % Line of sight signal x(t)
  delay(i) = los(i)*alpha; % delayed signal  alpha*x(t-tau)
  z(i) = x(i) + delay(i);  % z = x(t) + alpha*x(t-tau)
  t(i) = time(i+tau);;
end;

z = z+noise_fac*randn(size(z));  % add in noise

wt = w*time(tau+1);
wtmtau = w*time(1);
xkkm1=[ cos(wt)
       -w*sin(wt)
       alpha*cos(wtmtau)
       -w*alpha*sin(wtmtau)];


for (i=1:length(z))

          G=Pkkm1*C'*inv(C*Pkkm1*C' +R);
          Pkk=(I-G*C)*Pkkm1;
          Pkkm1=Phi*Pkk*Phi' + Q;
          e0=z(1,i)-[1  0   1    0]*xkkm1(:,i);
          e1=z(1,i)-[0 -1/w 1    0]*xkkm1(:,i);
          e2=z(1,i)-[0 -1/w 0 -1/w]*xkkm1(:,i);
   e3=z(1,i)-[1  0   0 -1/w]*xkkm1(:,i);
   %e_vect(:,i) = [ e0, e1, e2, e3 ]';

          if abs(e0)<=abs(e1) & abs(e0) <= abs(e2) % abs(e0)<=abs(e3); % t(i)<=s1;
          %if (t(i) <= switch1)
                  e=e0;
                  xkk(:,i) = xkkm1(:,i) +G*(e);
                  xkkps(1,i) = xkk(1,i);
                  xkkps(3,i) = xkk(3,i);
          elseif abs(e1) <= abs(e0) & abs(e1) <= abs(e2) & abs(e1)<=abs(e3) % s1 < t < s1+phi
          %elseif (switch1 <= t(i) & t(i) <= switch1+tau*dt)
                  e=e1;
                  xkk(:,i)=xkkm1(:,i)+G*(e);
                  xkkps(1,i) = -xkk(2,i)/w;
                  xkkps(3,i) = xkk(3,i);
   elseif abs(e2)<=abs(e0) & abs(e2)<=abs(e1) & abs(e2)<=abs(e3) % s1+tau < t < s2
   %elseif (switch1+tau*dt <= t(i) & t(i) <= switch2)
      e=e2;
      xkk(:,i)=xkkm1(:,i)+G*e;
      xkkps(1,i) = -xkk(2,i)/w;
      xkkps(3,i) = -xkk(4,i)/w;
   elseif abs(e3)<=abs(e0) & abs(e3)<=abs(e1) & abs(e3)<=abs(e2)
   %elseif (switch2 <= t(i) & t(i) <= switch2+tau*dt)
                                                            % s2 < t < s2+phi
      e=e3;
      xkk(:,i)=xkkm1(:,i)+G*e;
      xkkps(1,i)=xkk(1,i);
      xkkps(3,i)=-xkk(4,i)/w;
   else
          e=e0;
```

```
                xkk(:,i) = xkkm1(:,i) +G*(e);
                xkkps(1,i) = xkk(1,i);
                xkkps(3,i) = xkk(3,i);
   end;
            xkkm1(:,i+1)=Phi*xkk(:,i);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

figure;

mpath=delay;
los=x;
figure;

subplot(3,1,1);
plot(t,los,'r');
ylabel('los');
subplot(3,1,2);
plot(t,mpath,'b');
ylabel('mpath');

subplot(3,1,3);
plot(t,z,'g');
ylabel('z');

figure;
subplot(2,1,1)
plot(t,los,'r',t,xkkps(1,:),'b-.');
title('los & xkkps(1,:) (estimate = -.');

subplot(2,1,2);
plot(t,mpath,'r',t,xkkps(3,:),'b-.');
title('mpath & xkkps(3,:)');



% End of File mpath_ap.m
```

M. MY_GLB.M

```
% File Name: my_glb.m
% Last Updated:  16 Feb 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Global declaration for gps simulation and
% kalman filter.

global C;                     % Values initialized in kal_init.m
global Pkkml;
global R;
global Q;
global I;
global Phi;
global xkkml;

global w;

fl = 1;                       % NOTE:  These are the same as in init.m
To = 1;                       % and are placed here for use in non-gps model.
w=2*pi*fl;                    % Set frequency for simulation
phase_init = pi/6;            % Phase offset for sine wave in BPSK modulator
num_save = 100000;            % Number of points to save to workspace for plotting later

% End of File my_glb.m
```

N. SHIFT_REG.M

```
% File Name:  shift_reg.m
% Last Updated:  21 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description: This function generates the shift register
% linear code generators where R = the input values
%  size = register size.  The register is shifted right
%  and returned in a new vector

function r = shift_reg(R, size)

r = ones(1,size)


% End of File shift_reg.m
```

O. SIG_GEN.M

```
% File Name: sig_gen.m
% Last Updated:  11 Nov 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description: Returns either 1 P(Y) code xor with data
%                              2 P(Y) code
%                           or 3 C/A code xor with data
% the number is the sig_num which determines which signal
% is returned
%

function [s1,s2,s3] = sig_gen(code_length)

sv=1;
x1_epoch = 0;
x1A_epoch = 0;
x1B_epoch = 0;
x2A_epoch = 0;
x2B_epoch = 0;

%code_length = 10;

r_g1 = ones( size(1:10) ) ; % initialize register values
r_g2 = ones( size(1:10) );
r_x1a = [0 0 0 1 0 0 1 0 0 1 0 0];
r_x1b = [0 0 1 0 1 0 1 0 1 0 1 0];
r_x2a = [1 0 1 0 0 1 0 0 1 0 0 1];
r_x2b = [0 0 1 0 1 0 1 0 1 0 1 0];
shift_register = zeros(size(1:37));

t = 0;
data_clock = 20;

for i = 1:code_length
  if data_clock == 20
    data = 1;                % set to all ones to ignore
                data_clock = 0;
  else
                data_clock = data_clock + 1;
  end;

  for j = 1:10                              % gen C/A code at 1.023 MHz
        [r_g1, g_1] = g1(r_g1, x1_epoch);
        [r_g2, g_2] = g2(r_g2, x1_epoch, sv);
        g = xor(g_1, g_2);

        for k=1:10                                  % gen P/A code at 10.23 MHz
          t=t+1;                                    % or 10 * C/A code generation rate
          [r_x1a, A] = x1a(r_x1a, x1A_epoch);
          [r_x1b, B] = x1b(r_x1b, x1B_epoch);
          [r_x2a, C] = x2a(r_x2a, x2A_epoch);
          [r_x2b, D] = x2b(r_x2b, x2B_epoch);
          x1=xor(A,B);
          x2=xor(C,D);
          shift_register(2:37) = shift_register(1:36);
          shift_register(1) = x2;
          p=xor(x1, shift_register(sv));

          s1(t) = xor(p,data);
          s2(t) = p;
          s3(t) = xor(g, data);
        end; % for k;

  end; % for j

end; % for i



% End of File sig_gen.m
```

P. TEST_CA.M

```
% File Name: test_CA.m
% Last Updated: 31 Mar 97
% Written By: T. H. Newman
% Operating Environment: Windows 95
% Matlab Version 4.2c.1
%
% Description: Script file to test C/A code generation. Output
% should match First 10 C/A-Chips (Octal) of GPS-ICD-2000 table

for sv=1:37

r_g1 = round( rand( size(1:10) ) ); % initialize to some random state
r_g2 = round( rand( size(1:10) ) );
x1 = 1;
i=1;
[r_g1, g_1(i)] = g1(r_g1, x1);                % Initialize at epoch
[r_g2, g_2(i)] = g2(r_g2, x1, sv);
```

```
x1=0;
for i=2:10                                                    % Simulate 1st 10 chips
   [r_g1, g_1(i)] = g1(r_g1, x1);
   [r_g2, g_2(i)] = g2(r_g2, x1, sv);
end;

g = xor(g_1, g_2);                                    %

dig1 = g(1);
dig2 = g(2)*4 + g(3)*2 + g(4);                 % convert to octal number
dig3 = g(5)*4 + g(6)*2 + g(7);
dig4 = g(8)*4 + g(9)*2 + g(10);
result(sv) = dig1*10^3 + dig2*10^2 + dig3*10 + dig4;

end

result                                         % display results

% End of File test_CA.m
```

Q. TEST_P.M

```
% File Name:  test_P.m
% Last Updated:  1 April 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Script file to test P code generation.  Output
% should match First 12 P-Chips (Octal) of GPS-ICD-2000 table
% Note that only the first 13 sv's are calculated.  SV 9 - 37 are
% identical

for sv = 1:13
        r_x1a = round( rand( size(1:12) ) ); % initialize to some random values
        r_x1b = round( rand( size(1:12) ) );
        r_x2a = round( rand( size(1:12) ) );
        r_x2b = round( rand( size(1:12) ) );

        epoch = 1;
        i=1;
        [r_x1a, A(i)] = x1a(r_x1a, epoch);
        [r_x1b, B(i)] = x1b(r_x1b, epoch);
        [r_x2a, C(i)] = x2a(r_x2a, epoch);
        [r_x2b, D(i)] = x2b(r_x2b, epoch);

        epoch=0;
        for i=2:12
        [r_x1a, A(i)] = x1a(r_x1a, epoch);
        [r_x1b, B(i)] = x1b(r_x1b, epoch);
        [r_x2a, C(i)] = x2a(r_x2a, epoch);
        [r_x2b, D(i)] = x2b(r_x2b, epoch);
        end;

        x1=xor(A,B);
        x2=xor(C,D);
        delay=sv;
        x1=[x1, zeros(1,delay)];
        x2=[ones(1,delay), x2 ];
        p=xor(x1, x2);

        dig1 = p(1)*4 + p(2)*2 + p(3);
        dig2 = p(4)*4 + p(5)*2 + p(6);
        dig3 = p(7)*4 + p(8)*2 + p(9);
        dig4 = p(10)*4 + p(11)*2 + p(12);
        result(sv) = dig1*10^3 + dig2*10^2 + dig3*10 + dig4;
end; % for sv
result

% End of File test_P.m
```

R. UI_INIT.M

```
% File Name:  ui_init.m
% Last Updated:  4 May 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  Called by ui_vars.m to initialize parameters
%  after the Init button is pushed.
%


delay=zeros(size(1:max(tau) ));
if SNR > 0
  Noise_gain = sqrt((Amp^2)/( 10^(SNR/10) ));
else
  Noise_gain = 1;
end;

if SNR >=100
  Noise_gain = 0;
end;
w=2*pi*f;

disp(sprintf(' f=%2.2f  Initial Phase = %2.2f', f, p_degree))
disp(sprintf(' Amp=%2.2f', Amp))
disp(sprintf(' dt=%2.2f,  stop = %2.2f, Tb=%2.2f', dt, stop, Tb))
disp(sprintf(' SNR=%2.2f dB, Noise_gain=%2.2f',SNR, Noise_gain))
disp('tau='), disp(tau)
disp('alpha='),disp(alpha)


% End of File ui_init.m
```

S. UI_VARS.M

```
% File Name:  ui_vars.m
% Last Updated:  28 Apr 97
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  An Editable Text windows to control
% initialization values for kal_init
%        Calls: my_global.m to initialize values
%              kal_init.m  to initialize kalman filter variables
%              init.m to generate GPS code sample

global MainFig

% below are the initial values that will be in the text window
global C;                                % Values initialized in kal_init.m
global Pkkm1;
global R;
global Q;
global I;
global Phi;
global xkkm1;
global Tb;
global Noise_gain;
global dt;
global Amp;  % Amplitude for BPSK
global tau;
global alpha;
global delay;
global w;

global clock;

my_glb;

 % This initilizes w, phase_init, num_save, Tb, Noise_gain, A1, A2, dt

% Retreive  initial values from the strings
```

```
start = 0;            %start time
stop = 10;                    %stop time
f = w/(2*pi);                 %signal frequency
p_degree = phase_init*180/pi;      %phase in degrees
SNR = 100;

f_string = sprintf('      %2.2f',f);;
p_degree_string = sprintf('      %2.2f',p_degree);
Amp_string = sprintf('      %2.2f',Amp);
dt_string = sprintf('      %2.2f',dt);;
start_string = sprintf('      %2.2f',start);
stop_string = sprintf('      %2.2f',stop);
Tb_string = sprintf('      %2.2f',Tb);
SNR_string = sprintf('      %2.2f', SNR);

tau_string = '      [1 ]';
alpha_string = '      [1 ]';

tau=eval(tau_string);
alpha = eval(alpha_string);

%define the Main Figure Window.
num_entries = 9;
sep=45;
ymax = sep*(num_entries)+10;

x_pos=25; y_pos=35; dx=125; dy = ymax;

   MainFig = figure ('Position', [x_pos y_pos dx dy],...
    'Color', 'white','NumberTitle','off','Name',...
    'Simulation Variables',...
    'MenuBar','none');

%-----------------------------------------------%
% Define UIMENU for pull down menu options

e=uimenu('label','Init','callback',...
        'kal_init,ui_init; ');

f=uimenu('label', 'Plot');
  uimenu(f, 'label','Phase Lock Loop','callback','plot_k');
  uimenu(f,'label','Delay','callback','plot_d');
  uimenu(f,'label','QPSK','callback','plot_q2');
  uimenu(f,'label','BPSK','callback','plotbpsk');
  uimenu(f,'label','Error','callback','plot_err');
  uimenu(f,'label','Variables','callback','plot_k2');

g=uimenu('label', 'Sim');
  g1=uimenu(g, 'label','BPSK');
    uimenu(g1, 'label', 'BPSK Simulation','callback','kal_sim');
    uimenu(g1, 'label', 'PLL Simulation','callback','pll');
  g2=uimenu(g, 'label', 'QPSK');
```

```
      uimenu(g2, 'label','QPSK Simulation', 'callback','kalq_sim');
      uimenu(g2, 'label', 'PLL Simulation','callback','pll_q');
   g3=uimenu(g, 'label', 'Unbalanced QPSK');
      uimenu(g3, 'label','Unbalanced QPSK Simulation', 'callback','kalu_sim');
      uimenu(g3, 'label', 'PLL Simulation','callback','pll_u');
   g4=uimenu(g,'label','Multipath','callback','mpathsim');



 n=1;  % ui menu  number
 ymax = dy;
 xpos=10;
 dx = 100;
 dy=20;
 space = 22;

 label_fg_color = 'Black';
 label_bg_color = 'White';
 box_fg_color = 'Black';
 box_bg_color = 'Yellow';

 f_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
     'String', 'Signal freq (Hz)','ForegroundColor',label_fg_color,...
     'BackgroundColor',label_bg_color);


 f_box = uicontrol(MainFig,'Style','edit','String', f_string,...
     'Pos',[xpos ymax-n*sep dx dy],...
     'Foregroundcolor',box_fg_color,...
     'BackgroundColor',box_bg_color,...
     'Callback','f = eval( get(f_box,"String") ); w=2*pi*f;');

n=n+1;

phase_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
     'String', 'Phase Offset °','ForegroundColor',label_fg_color,...
     'BackgroundColor',label_bg_color);

phase_box = uicontrol(MainFig,'Style','edit','String', p_degree_string,...
     'Pos',[xpos ymax-n*sep dx dy],...
     'Foregroundcolor',box_fg_color,...
     'BackgroundColor',box_bg_color,...
     'Callback','p_degree = eval( get(phase_box,"String") );phase_init=p_degree*pi/180');

n=n+1;

amp_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
     'String', 'Amplitude','ForegroundColor',label_fg_color,...
     'BackgroundColor',label_bg_color);

amp_box = uicontrol(MainFig,'Style','edit','String', Amp_string,...
     'Pos',[xpos ymax-n*sep dx dy],...
```

```
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','A1 = eval( get(amp_box,"String") );');

n=n+1;

samp_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'Ts (sec)','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

samp_box = uicontrol(MainFig,'Style','edit','String', dt_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','dt = eval( get(samp_box,"String") );');

n=n+1;

stop_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'Stop time (sec)','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

stop_box = uicontrol(MainFig,'Style','edit','String', stop_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','stop = eval ( get(stop_box,"String") );');

n=n+1;

Tb_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'Tb (sec)','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

Tb_box = uicontrol(MainFig,'Style','edit','String', Tb_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','Tb = eval ( get(Tb_box,"String") );');

n=n+1;

SNR_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'SNR dB','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

SNR_box = uicontrol(MainFig,'Style','edit','String', SNR_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','SNR = eval( get(SNR_box,"String") ); ');
```

```
n=n+1;

tau_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'tau','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

tau_box = uicontrol(MainFig, 'Style','edit','String', tau_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','tau = eval( get(tau_box,"String") ); ');

n=n+1;


alpha_label = uicontrol(MainFig, 'Style','text', 'Pos',[xpos ymax-n*sep+space dx dy],...
    'String', 'alpha','ForegroundColor',label_fg_color,...
    'BackgroundColor',label_bg_color);

alpha_box = uicontrol(MainFig,'Style','edit','String', alpha_string,...
    'Pos',[xpos ymax-n*sep dx dy],...
    'Foregroundcolor',box_fg_color,...
    'BackgroundColor',box_bg_color,...
    'Callback','alpha = eval( get(alpha_box,"String") ); ');

n=n+1;
%-----------------------------------------------%
% Define the UICONTROL button for callbacks.
% b1=uicontrol('style','pushbutton','string','Initialize',...
%    'position',[30 ymax-n*sep 65 25],'Callback',...
%    'kal_init; delay=zeros(size(1:max(tau) ));');



% End of File ui_vars.m
```

T. X1A.M

```
% File Name: x1a.m
% Last Updated: 18 Oct 96
% Written By: T. H. Newman
% Operating Environment: Windows 95
% Matlab Version 4.2c.1
%
```

```
% Description:  This function generates the x1a(t) signal
% for GPS P Code generator

function [new_R, A] = x1a(R, x1a_epoch)

if x1a_epoch==1
  R = [0 0 0 1 0 0 1 0 0 1 0 0];  % Reset at X1A epoch
end;
new_R = gen_poly([1,6,8,11,12], R);

A=R(12);

% End of File x1a.m
```

U.   X1B.M

```
% File Name:  x1b.m
% Last Updated:  18 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  This function generates the x1b(t) signal
% for GPS P Code generator

function [new_R, B] = x1b(R, x1b_epoch)

if x1b_epoch==1
  R = [0 0 1 0 1 0 1 0 1 0 1 0];  % Reset at X1B epoch
end;
new_R = gen_poly([1 1 2 5 8 9 10 11 12], R);

B=R(12);

% End of File x1b.m
```

V.   X2A.M

```
% File Name:  x2a.m
% Last Updated:  18 Oct 96
% Written By:  T. H. Newman
% Operating Environment:  Windows 95
% Matlab Version 4.2c.1
%
% Description:  This function generates the x2a(t) signal
% for GPS P/A Code generator
```

```
function [new_R, C] = x2a(R, x2a_epoch)

if x2a_epoch==1
  R = [1 0 1 0 0 1 0 0 1 0 0 1]; % Reset at X2a epoch
end;
new_R = gen_poly([1 1 3 4 5 7 8 9 10 11 12], R);

C=R(12);

% End of File x2a.m
```

W. X2B.M

```
% File Name: x2b.m
% Last Updated: 18 Oct 96
% Written By: T. H. Newman
% Operating Environment: Windows 95
% Matlab Version 4.2c.1
%
% Description: This function generates the x2b(t) signal
% for GPS P Code generator

function [new_R, D] = x2b(R, x2b_epoch)

if x2b_epoch==1
  R = [0 0 1 0 1 0 1 0 1 0 1 0]; % Reset at X2B epoch
end;
new_R = gen_poly([1 2 3 4 8 9 12], R);

D=R(12);

% End of File x2b.m
```

APPENDIX B SIMULINK DIAGRAMS

A. BPSK.M

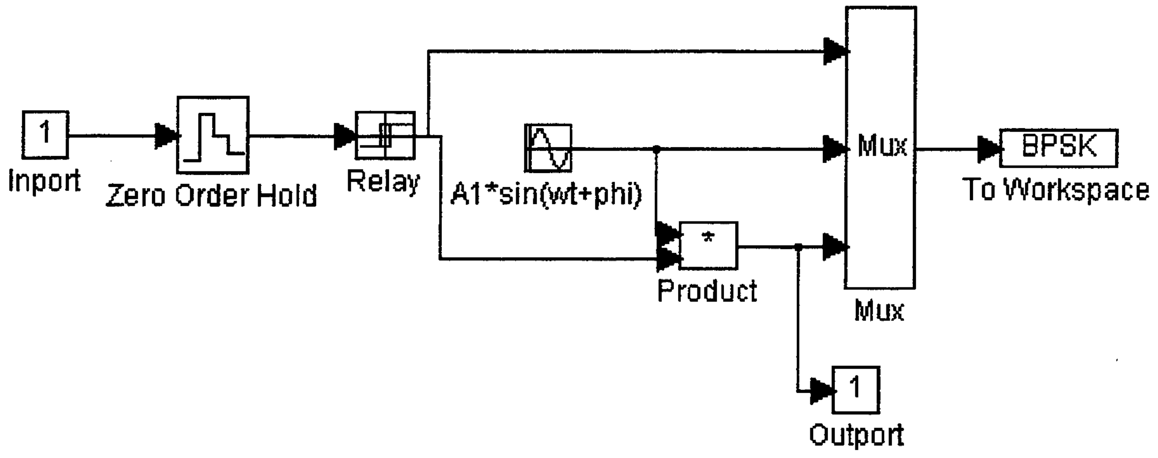


Figure 35 BPSK.M

B. BPSK_P90.M

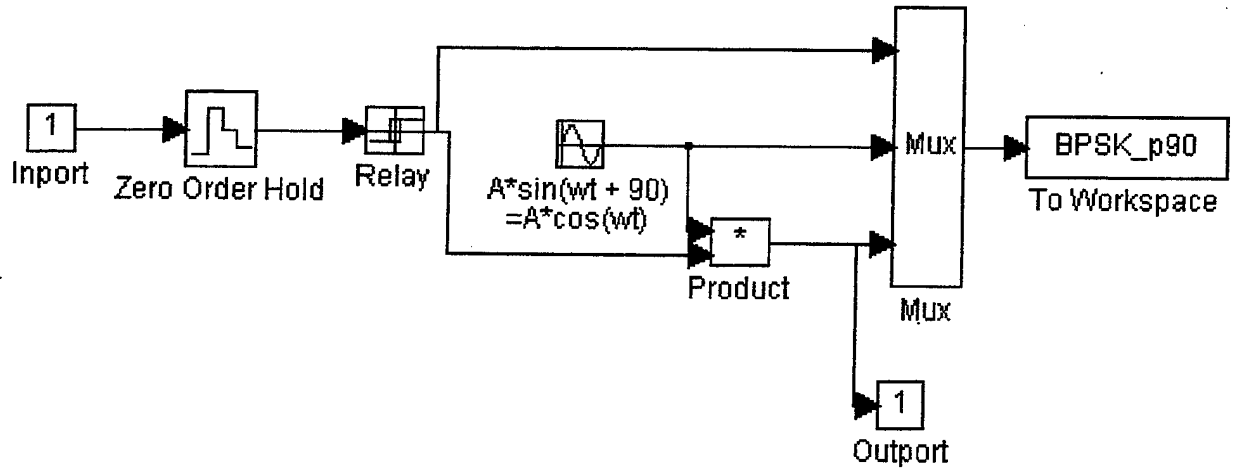


Figure 36 BPSK_P90.M

C. BPSK_U.M

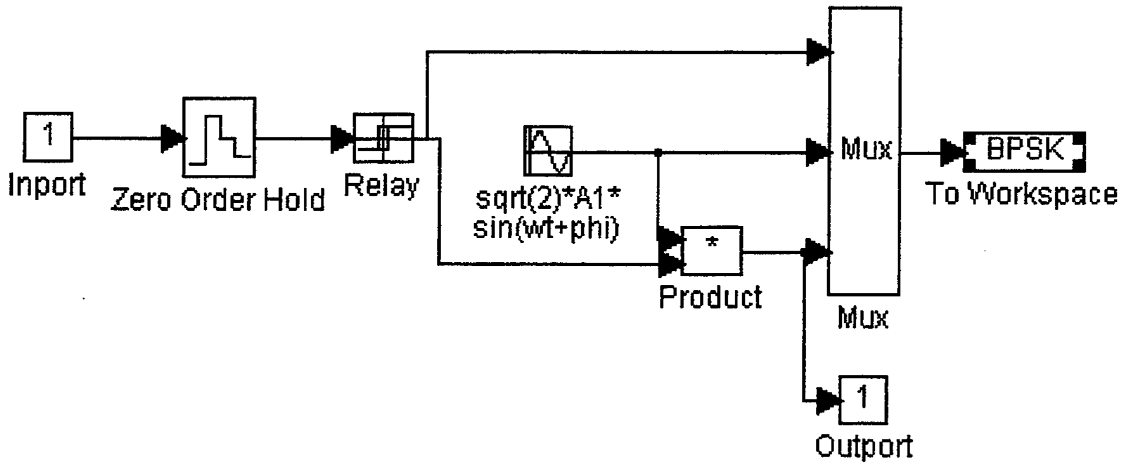


Figure 37 BPSK_U.M

D. KAL_SIM.M

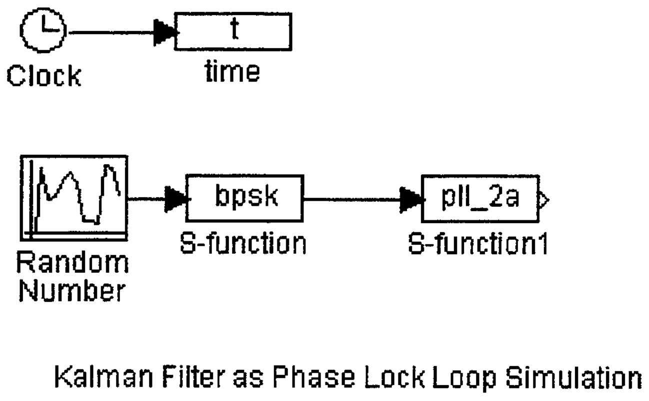


Figure 38 KAL_SIM.M

E. KALQ_SIM.M

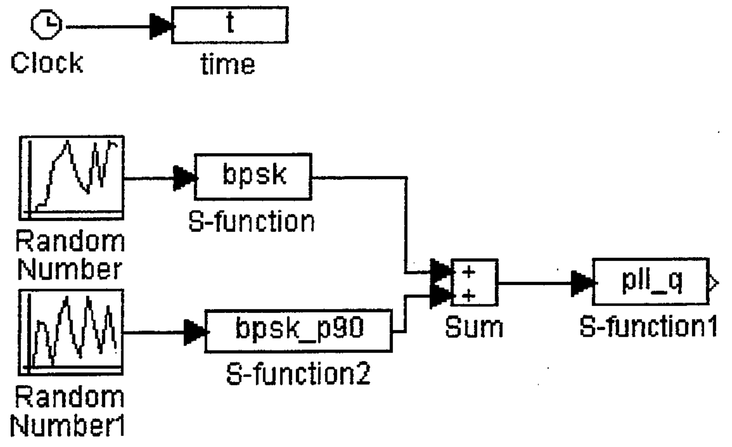


Figure 39 KALQ_SIM.M

F. KALU_SIM.M

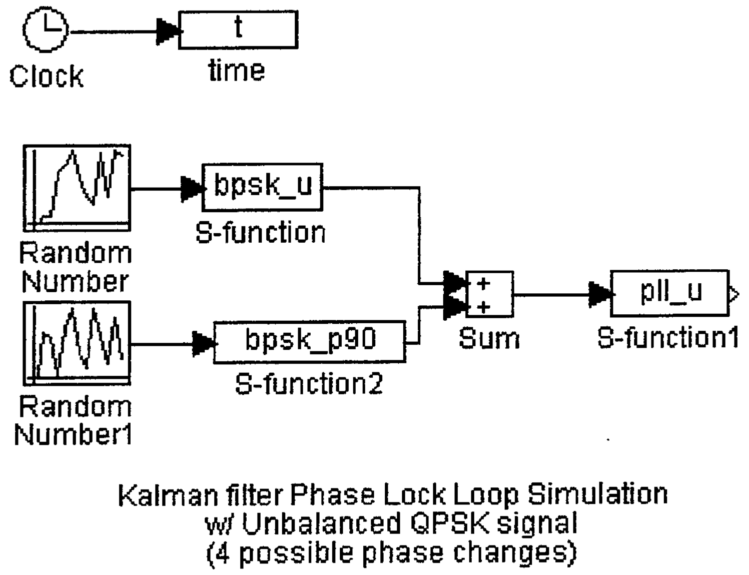
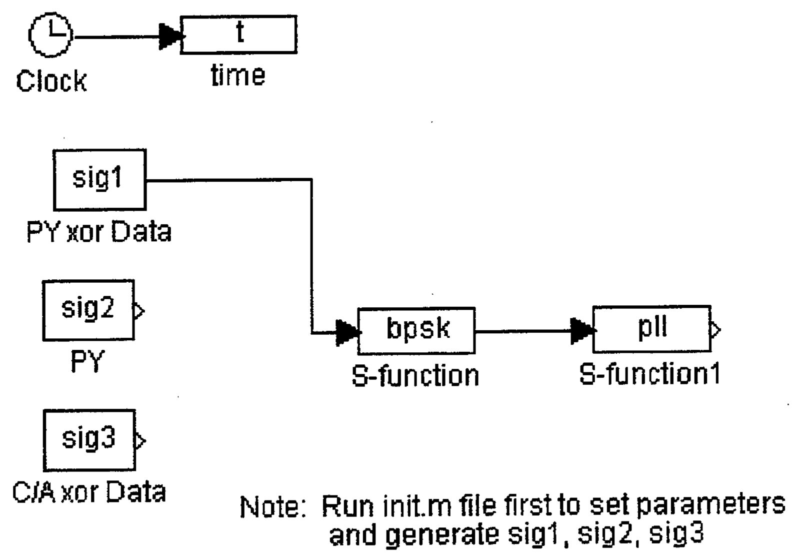


Figure 40 KALU_SIM.M

G. L1_SIM



Clock
t
time
sig1
PY xor Data
sig2
PY
sig3
C/A xor Data
bpsk
S-function
pll
S-function1
Note: Run init.m file first to set parameters and generate sig1, sig2, sig3

Figure 41 L1_SIM.M

H. L2_SIM.M

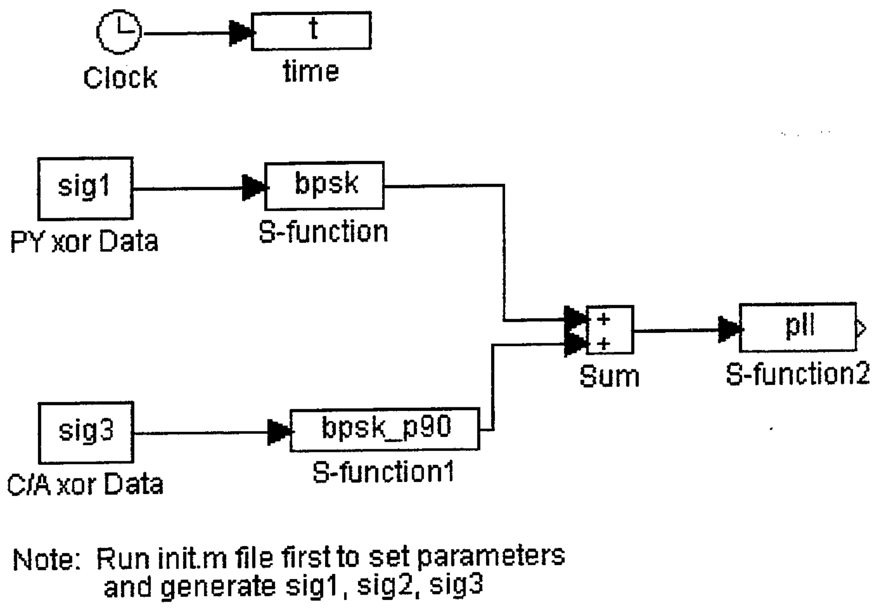


Figure 42   L2_SIM.M

I. MPATHSIM.M

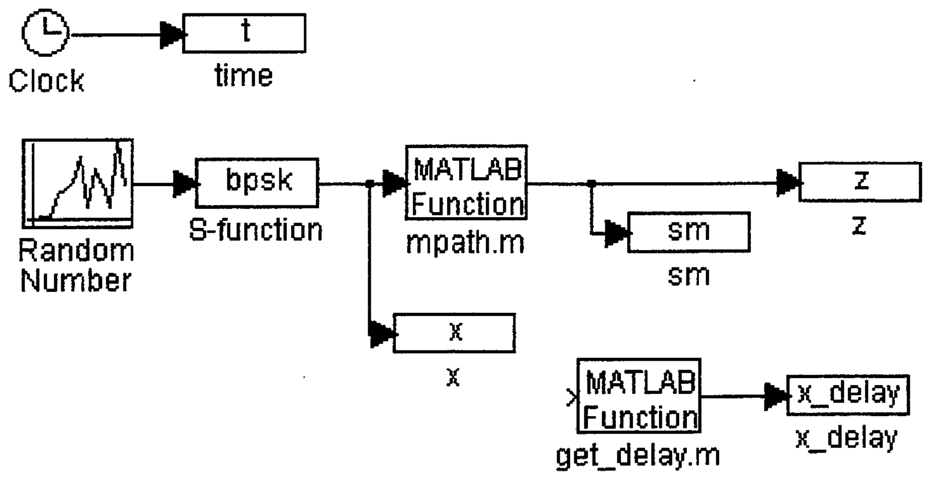


Figure 43 MPATHSIM.M

J. PLL.M

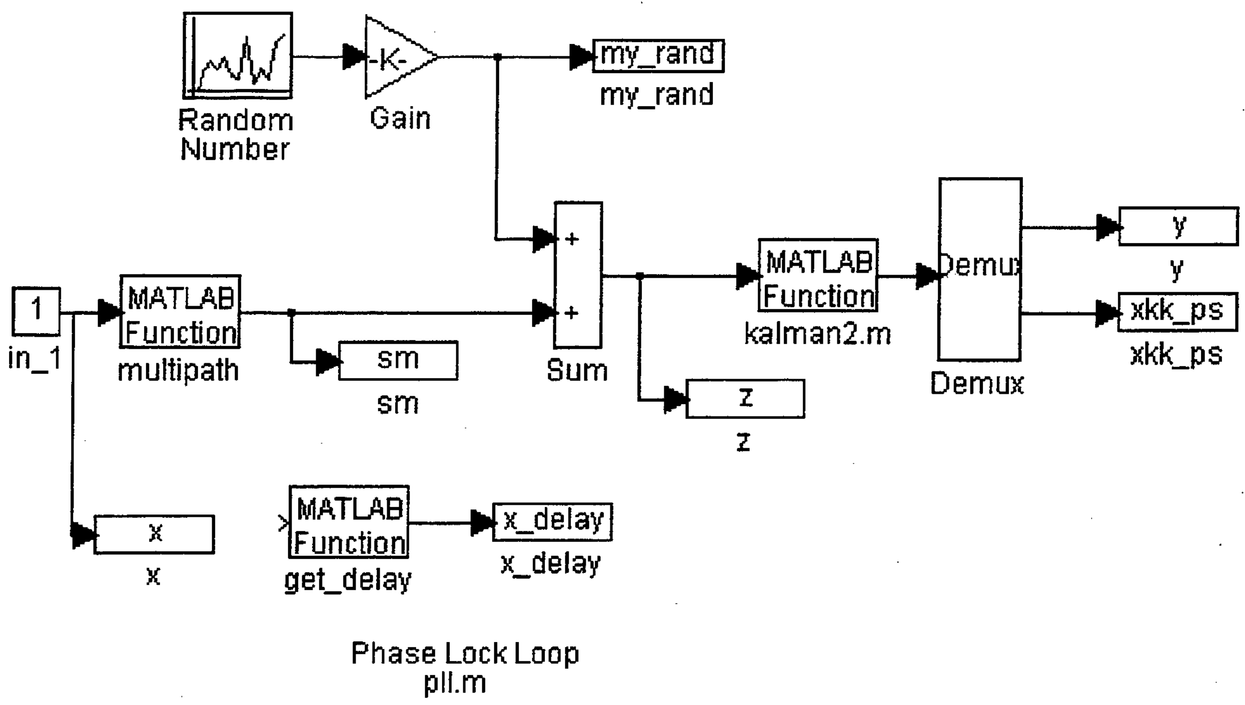


Figure 44 PLL.M

K. PLL_2A.M

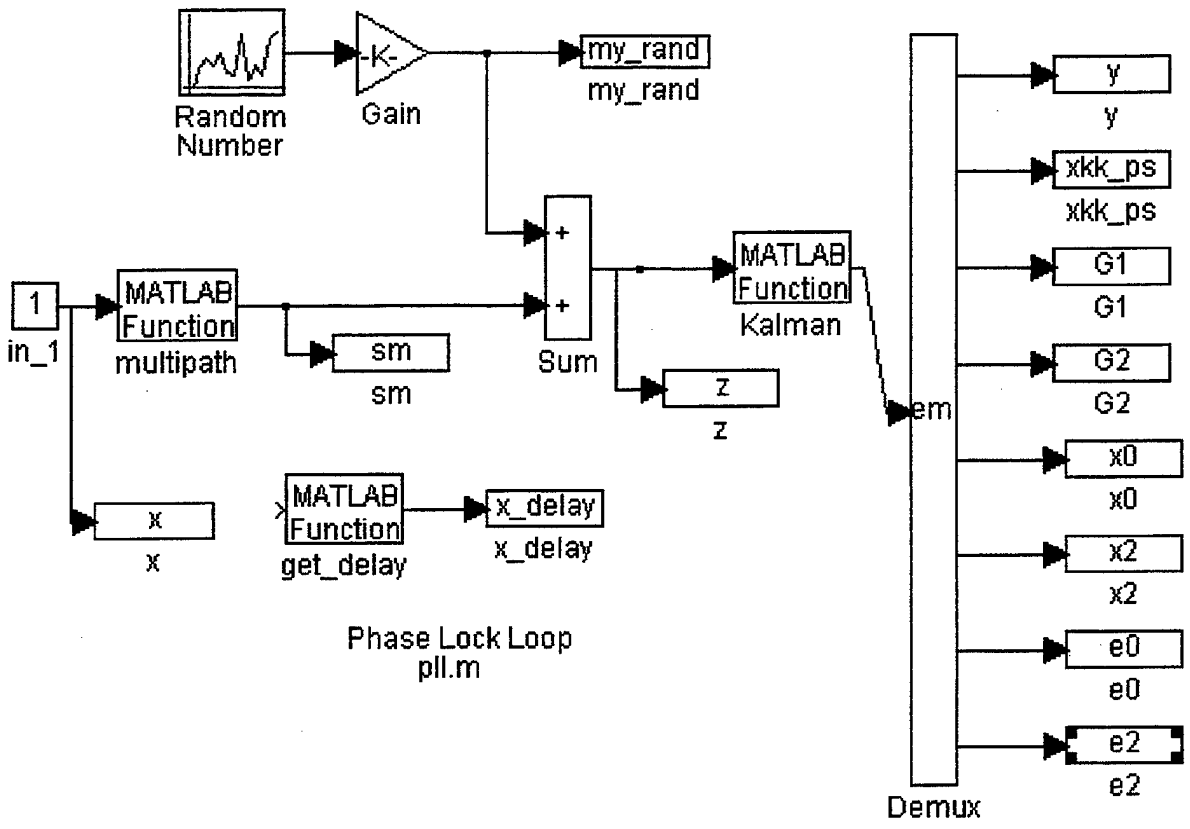


Figure 45 PLL2A.M

APPENDIX C EXAMPLE SNR CALCULATION

$dB2R(x) := 10^{\frac{x}{10}}$ convert dB to ratio $dB(x) := 10 \cdot \log(x)$ convert ratio to dB

Minimum Received Power L1 with C/A Code	$C := -160.0$	~ dB_W
Boltzman's Constant	$k := -228.6$	~ dB W/Hz-K
Noise Temperature	$Tn := 28$	~ dB K
Received Noise	$No := k + Tn$ $No = -200.6$	~ dB W/Hz
Carrier to Noise Density Ratio	$CNR := C - No$ $CNR = 40.6$	~ db Hz
Chip Data Rate	$R := 1.023 \cdot 10^{6}$	~ chips/sec
	$R\_db := dB(R)$ $R\_db = 60.099$	~ dB cps
Information Data Rate	$R\_i := 50$	~ bits/sec
Band Width - QPSK	$B := 0.6 \cdot R$	
	$B\_dB := dB(B)$	
	$B\_dB = 57.88$	~ dB bps
Signal to Noise Ratio	$SNR := CNR - B\_dB$ $SNR = -17.28$	~ dB
Processing Gain	$PG := dB\left(\frac{R}{R\_i}\right)$ $PG = 43.109$	~ dB
Signal to Noise Ratio	$SNR := SNR + PG$ $SNR = 25.829$	~ dB

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code EC 1
   833 Dyer Rd., Room 437
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Space Systems Academic Group, Code SP 1
   777 Dyer Rd., Rm. 200
   Naval Postgraduate School
   Monterey, CA 93943-5110

5. Professor Hal Titus, Code EC/Ts...1
   833 Dyer Rd., Room 437
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Commander Gus Lott, Code EC/Lt..1
   833 Dyer Rd., Room 437
   Naval Postgraduate School
   Monterey, CA 93943 - 5121

7. Commanding Officer...1
   (Attn: Code 30, CDR Zellmann)
   Naval Information Warfare Activity
   9800 Savage Road
   FT Meade, MD 20755-6000

8. Commanding Officer...1
   Naval Command, Control and Ocean Surveillance Center
   RDT&E Division Code D841
   Attn: Dr. Gerry Baumgartner
   49275 Electron Drive
   San Diego, CA 92152-5435

9. Jet Propulsion Laboratory..1
(Attn: Kenneth H. Lau, Mail Stop 198-326)
4800 Oak Grove Drive
Pasadena, CA 91109-8099

10. Director..1
National Security Agency
Attn: J6 (Fran Landolf)
Fort George G. Meade, MD 20755-6000

11. Director..1
National Security Agency
Attn: N5 (John Wibbe)
Fort George G. Meade, MD 20755-6000

12. Director..1
National Security Agency
Attn: D (Dr. John O'Hara)
Fort George G. Meade, MD 20755-6000

13 LT Thomas Newman..1
389-H Ricketts Road
Monterey, CA 93940